STM32 mikrovezérlők programozása STM32CubeIDE környezetben

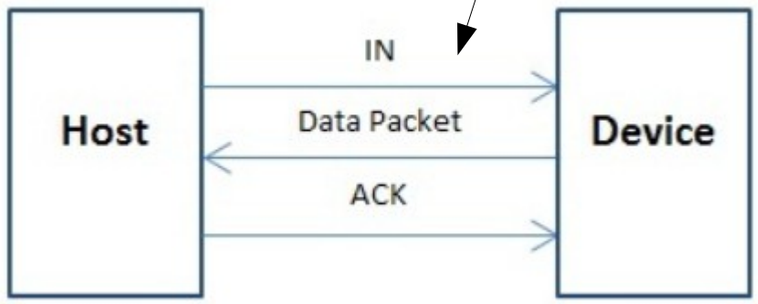
13. Az USB tömegtároló eszközosztály (MSC)

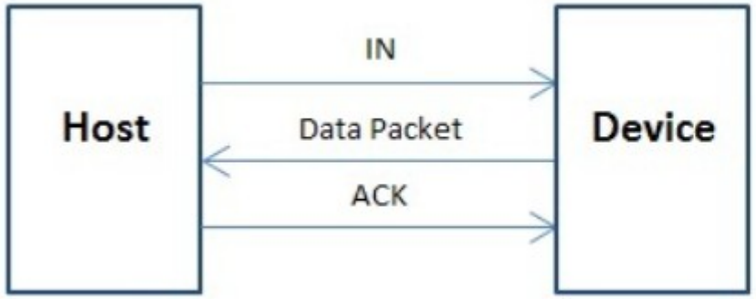
Felhasznált és ajánlott irodalom

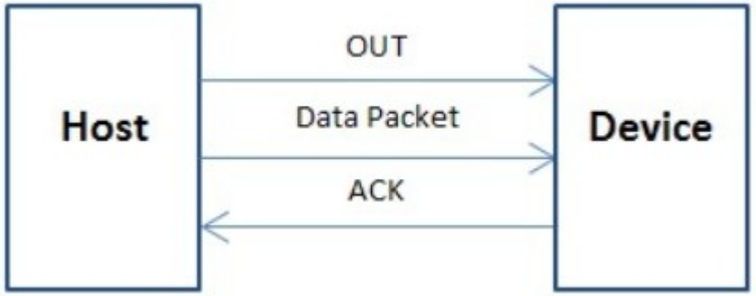
- STMicroelectronics: STM32 USB training (27 video + letölthető anyagok)
  - Youtube lejátszási lista
  - Letölthető anyagok (ZIP fájl)
- STMicroelectronics: UM1734 - STM32Cube™ USB device library
- Jan Axelson: USB complete (könyv)
- Jan Axelson: USB Mass Storage (könyv)
- USB-IF: USB 2.0 Specification
- USB-IF: USB MSC specification overview
- USB-IF: Mass Storage Bulk Only 1.0
- ControllersTech: STM32 USB MSC

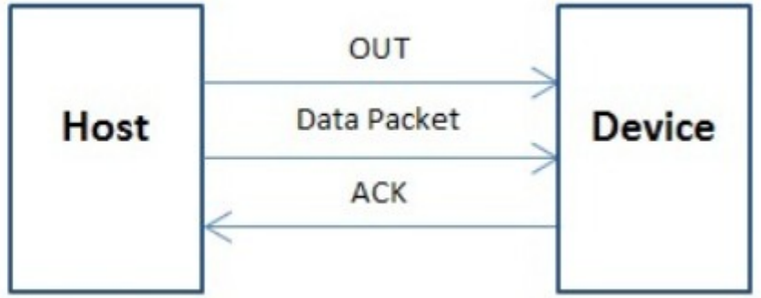
Az USB tömegtároló osztály

- A tömegtároló osztályba (USB MSC) azok az eszközök tartoznak, amelyek fájlok átvitelére szolgálnak és (merev)lemez, CD, DVD vagy flash memória tárolót tartalmaznak
- Windows Intéző alatt a tömegtároló eszközök lemezmeghajtóként jelennek meg
- A tömegtároló eszközök általában **bulk átviteli módot használnak**
- A **Bulk Only Transfer** (BOT) eszközök alapvetően **bulk átviteli móddal** kommunikálnak, működésük módja két, vagy három fokozatú:
  - A host egy parancsblokkot küld az eszköznek
  - Adatblokk(ok) küldése vagy fogadása (ha kell)
  - Az eszköz státuszinformációt küld a hostnak









CBW – Command Blok Wrapper

- A host a parancsot egy 31 bájt hosszúságú **CBW** struktúrába csomagolva küldi, melynek formátuma az alábbi táblázatban látható *(forrás: Jan Axelson – USB complete)*

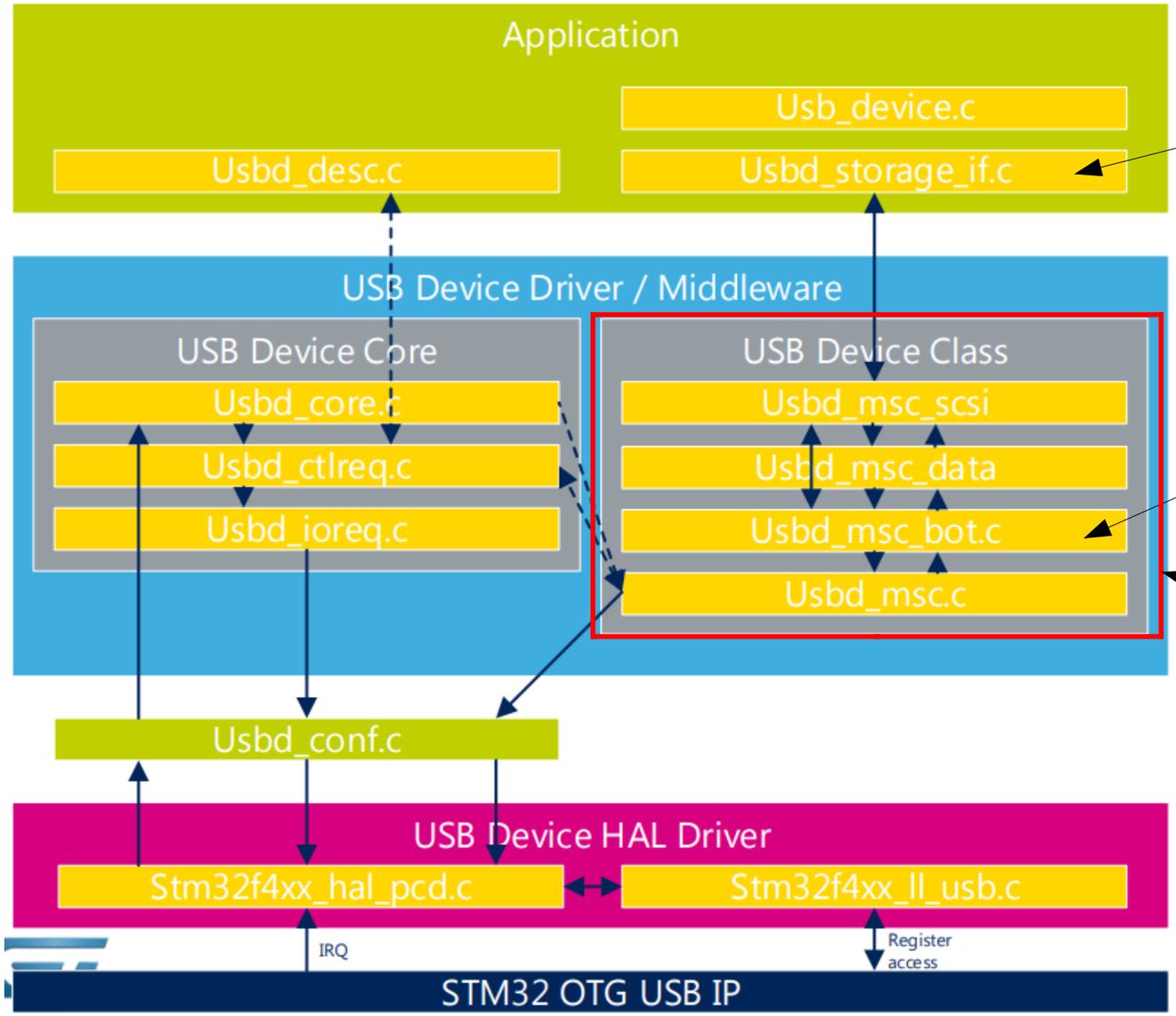
Name	Bits	Description
dCBWSignature	32	The value 43425355h, which identifies the structure as a CBW.
dCBWTag	32	A tag that associates this CBW with the CSW the device will send in response.
dCBWDataTransferLength	32	The number of bytes the host expects to transfer in the data-transport stage.
bmCBWFlags	8	Specifies the direction of the data-transport stage. Bit 7 = 0 for an OUT (host-to-device) transfer. Bit 7 = 1 for an IN (device-to-host) transfer. All other bits are zero. If there is no data-transport stage, bit 7 is ignored.
Reserved	4	Zero
bCBWLUN	4	For devices with multiple LUNs, specifies the LUN the command block is directed to. Otherwise the value is zero.
Reserved	3	Zero
bCBWCBLength	5	The length (1–16) of the command block in bytes
CBWCB	128	The command block for the device to execute.

CSW – Command Status Wrapper

- A státusz átvitelekor az eszköz egy 1b bájt hosszúságú **CSW** struktúrába csomagolva küldi, melynek formátuma az alábbi táblázatban látható *(forrás: Jan Axelson – USB complete)*
- A **CBW** érvényességét az eszköznek, a **CSW** érvényességét pedig a hostnak kell ellenőriznie *(hosszméret, signature, statusz esetén a Tag egyezése, parancs esetén hogy reset vagy státuszjelentés után érkezett-e)*

Name	Bits	Description
dCBWSignature	32	The value 53425355h, which identifies the structure as a CSW.
dCBWTag	32	The value of the dCBWTag in a CBW received from the host.
dCSWDataResidue	32	For OUT transfers, the difference between dCBWDataTransferLength and the number of bytes the device processed. For IN transfers, the difference between dCBWDataTransferLength and the number of bytes the device sent.
bCSWStatus	8	00h = command passed 01h = command failed 02h = phase error

STM32 USB MSC eszköz szoftver struktúrája

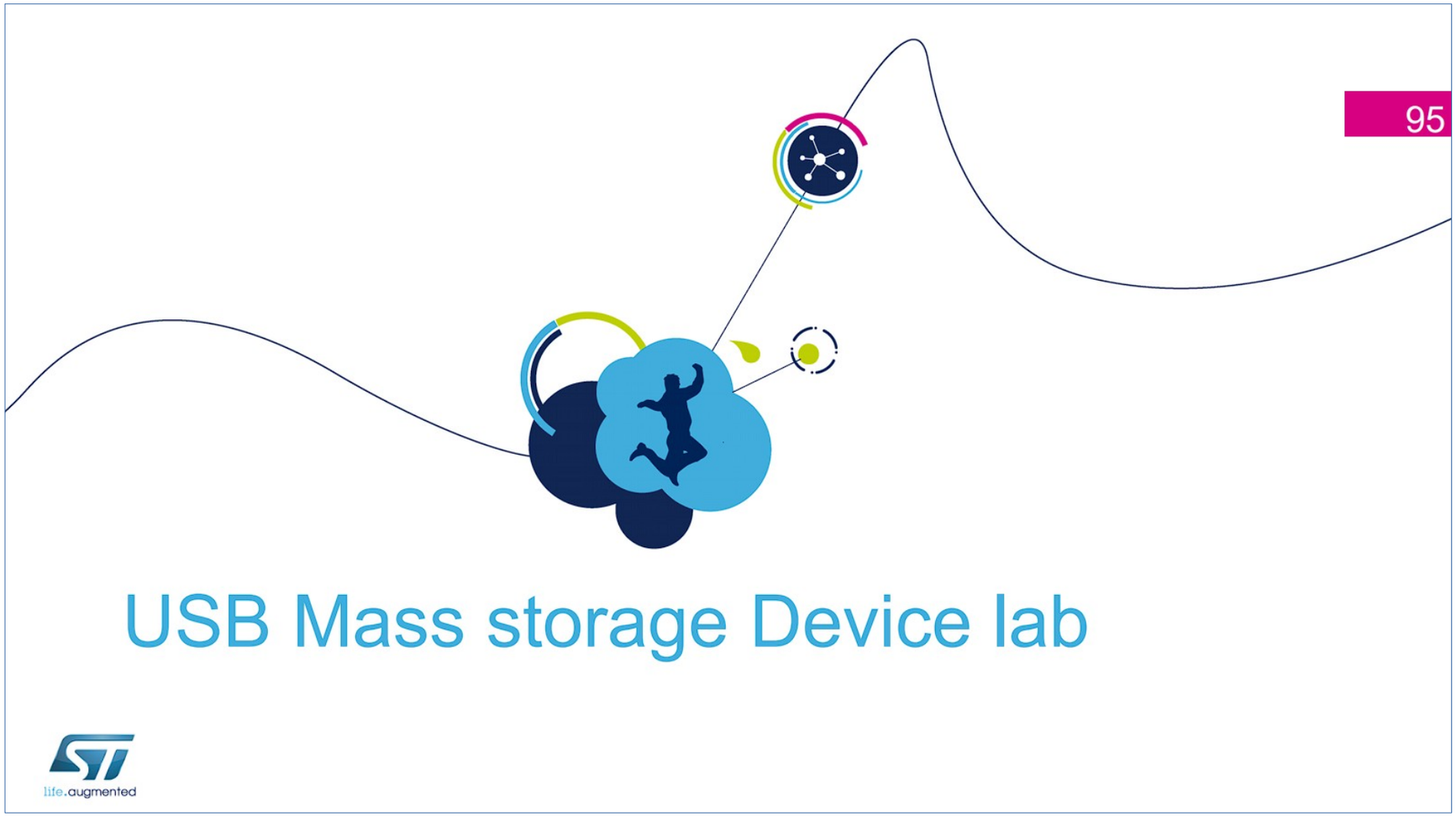


- Alkalmazásprogramozói interfész
- A **BOT** eszközkezelés implementálása
- Osztályspecifikus firmware kód

(Middlewares\ST\STM32_USB_Device_Library\Class)

USB MSC device lab

- Az STMicroelectronics: *"STM32 USB training"* c. tanfolyamának **16. videója** egy **USB MSC** demóprogramot mutat be, ebből fogunk kiindulni és ezt fogjuk továbbfejleszteni.



Példaprogramok

Nucleo-F446RE kártya:

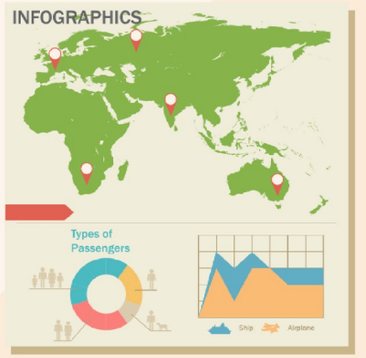
F446RE_USB_MSC – USB mass storage demó (RAM)

F446RE_USB_MSC_FRAM – USB mass storage FRAM-mal

F446RE_USB_MSC_SDIO – USB SD kártyaolvasó

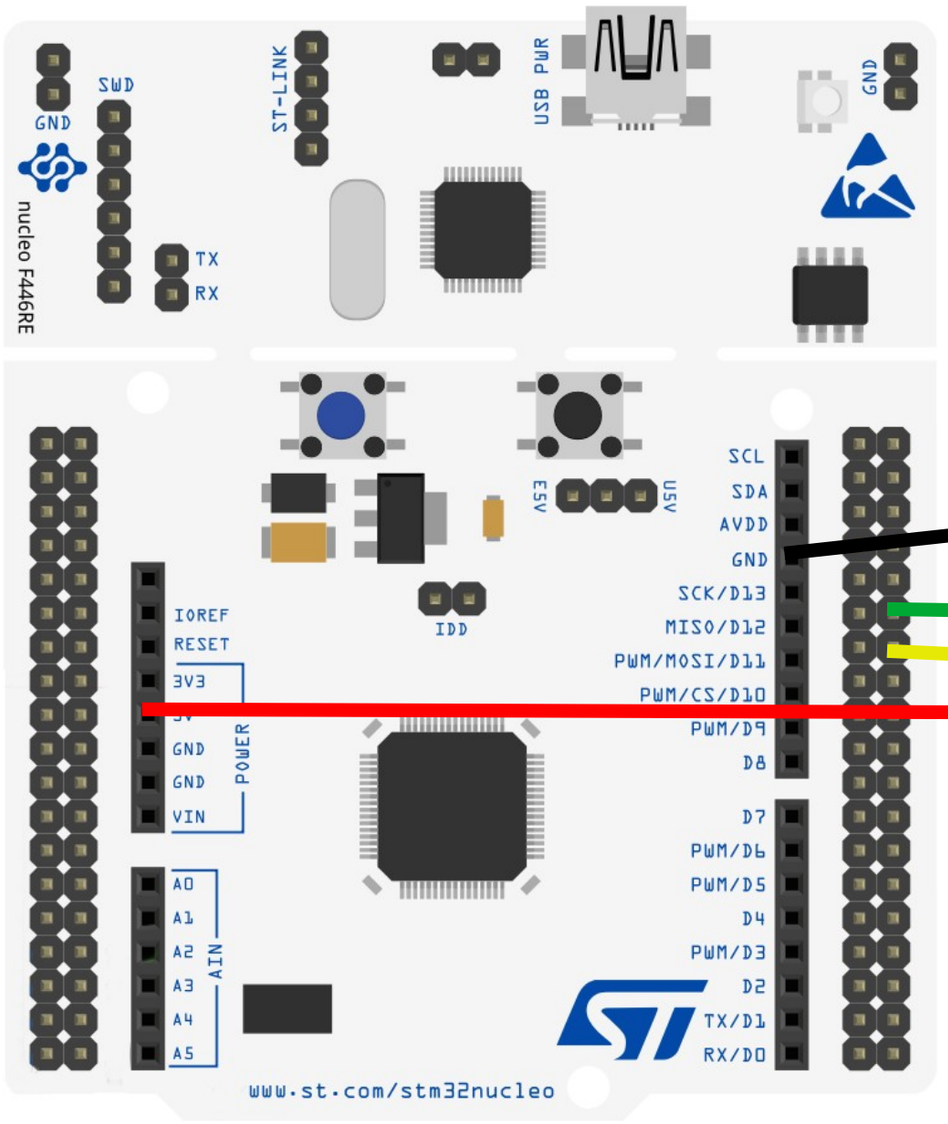
Blue pill kártya:

F103_USB_MSC_FRAM – USB mass storage FRAM-mal

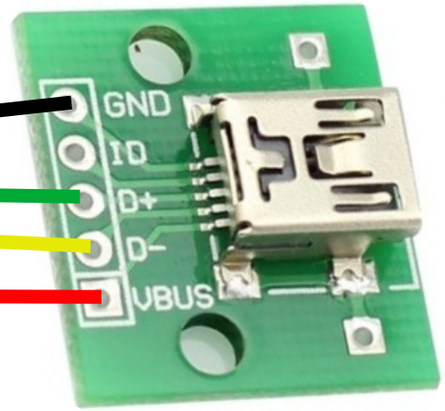


freepik.com

Bekötési vázlat

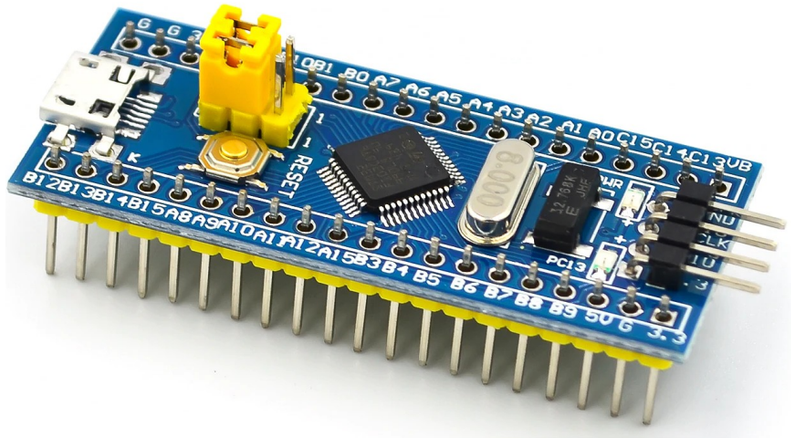


PA12 USB D+
PA11 USB D-



STM32F103C8: a **D+** vonalat 1,5 kΩ-mal 3.3 V-ra fel kell húzni!

STM32F4xx: a **D+** vonal külső felhúzására nincs szükség!



F446RE_USB_MSC: USB konfigurálás

- USB OTG FS = ***Device only*** és ***Mass Storage Class*** aktiválás kell

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

Categories | A->Z

System Core
- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog

Timers

Connectivity

Multimedia

Computing

Middleware
- FATFS
- FREERTOS
- LIBJPEG
- MBEDTLS
- PDM2PCM
- USB_DEVICE
- USB_HOST

USB_DEVICE Mode and Configuration

Mode

Class For HS IP: Disable

Class For FS IP: Mass Storage Class

Configuration

Reset Configuration

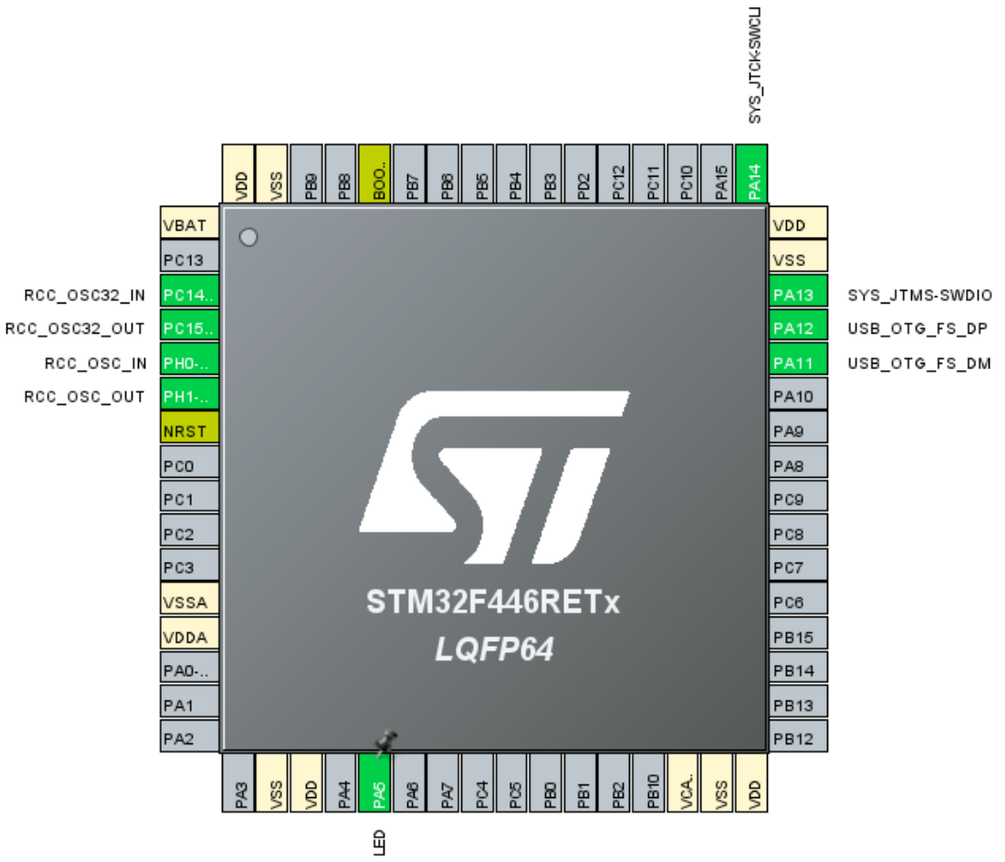
Parameter Settings | Device Descriptor | User Constants

Configure the below parameters :

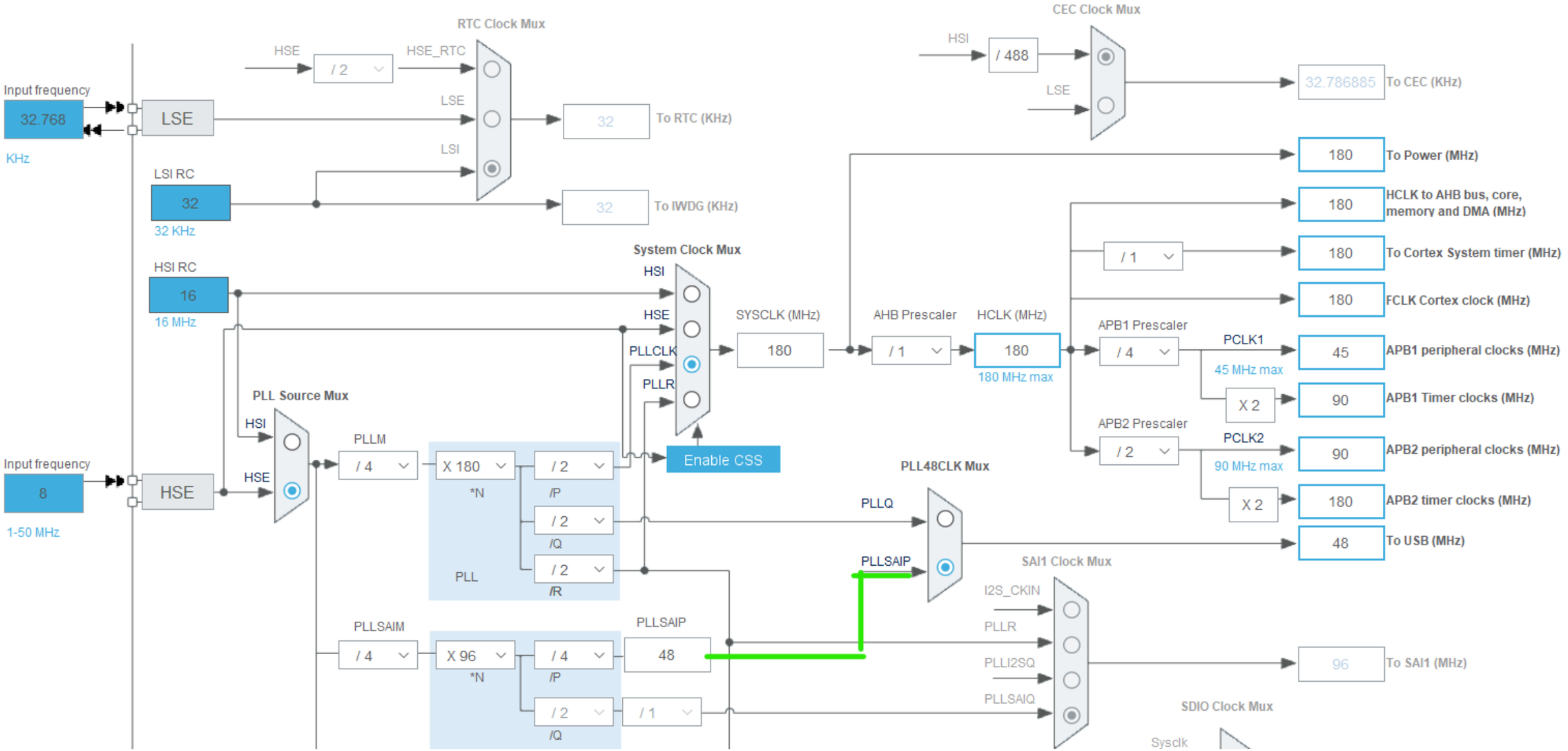
Search (Crtl+F)

Basic Parameters	
USBD_MAX_NUM_INTERFACES (Maximum number of supported interfaces)	1
USBD_MAX_NUM_CONFIGURATION (Maximum number of supported configuration)	1
USBD_MAX_STR_DESC_SIZ (Maximum size for the string descriptors)	512 bytes
USBD_SELF_POWERED (Enabled self power)	Disabled
USBD_DEBUG_LEVEL (USBD Debug Level)	0: No debug message
USBD_LPM_ENABLED (Link Power Management)	0: Link Power Management not ...
Class Parameters	
MSC_MEDIA_PACKET (Media I/O buffer Size)	512 bytes

Pinout view | System view

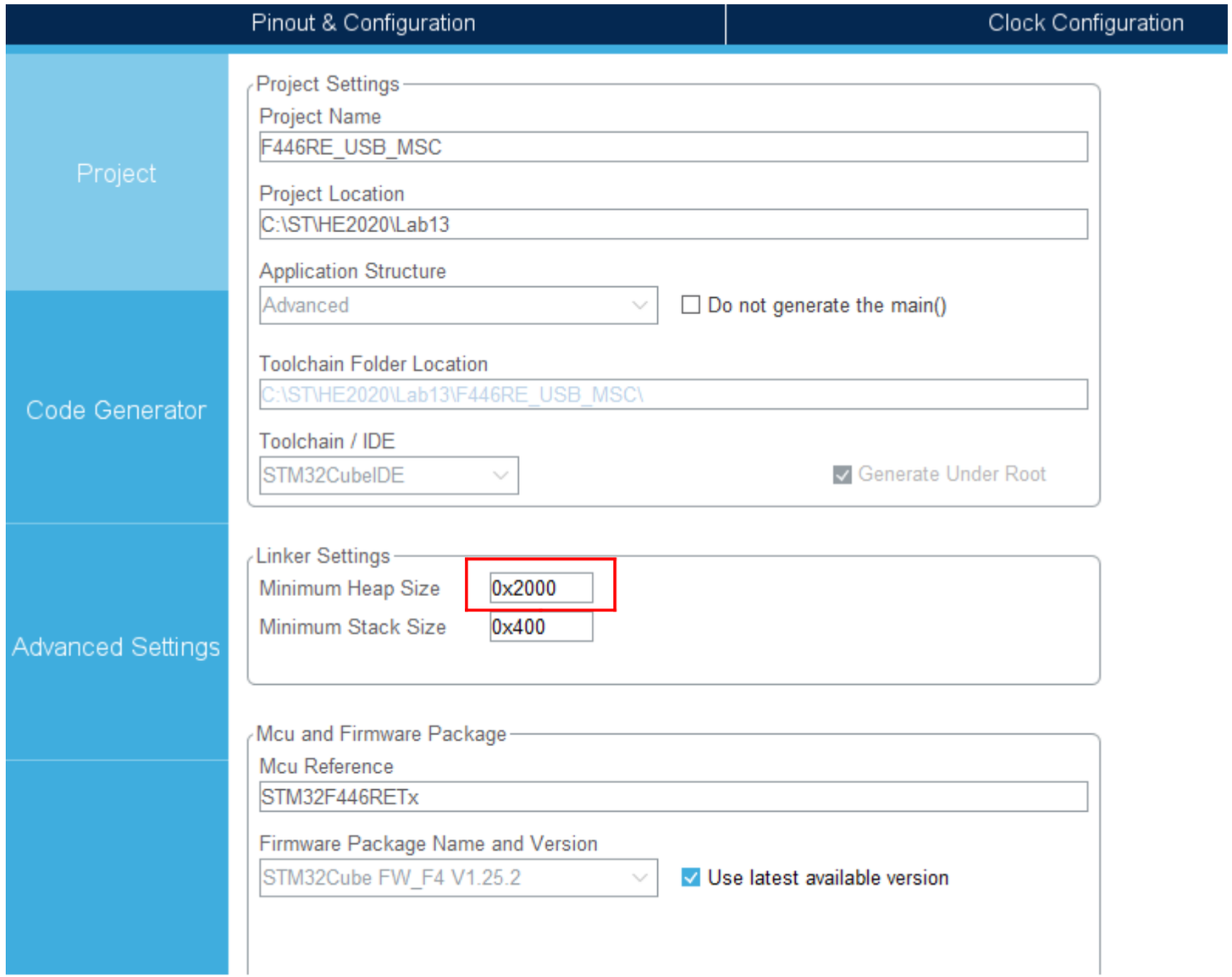


F446RE_USB_MSC: órajel konfigurálás



F446RE_USB_MSC: projekt konfigurálás

- A ***Minimum Heap Size*** paraméter értékét meg kell növelni, 0x2000 elegendő lesz



F446RE_USB_MSC: usbd_storage_if.c módosítása

- Kódgenerálás után csak az **usbd_storage_if.c** forrásállományt kell módosítani. Az alábbiakban az STM32 USB training - 09.8 USB MSC device labs útmutatását követjük:
- A fájl elején módosítjuk a blokkok számát és létrehozzuk a RAM buffert

```
#define STORAGE_LUN_NBR                  1
#define STORAGE_BLK_NBR                  0x80      //--- Modified!!!
#define STORAGE_BLK_SIZ                  0x200

/* USER CODE BEGIN PRIVATE_DEFINES */
uint8_t buffer[STORAGE_BLK_NBR*STORAGE_BLK_SIZ];   //--- User data will be stored here
/* USER CODE END PRIVATE_DEFINES */
```

- Az interfész függvények közül a **STORAGE_Read_FS()** és a **STORAGE_Write_FS()** függvényeket kell átszabni

```
static int8_t STORAGE_Init_FS(uint8_t lun);
static int8_t STORAGE_GetCapacity_FS(uint8_t lun, uint32_t *block_num, uint16_t *block_size);
static int8_t STORAGE_IsReady_FS(uint8_t lun);
static int8_t STORAGE_IsWriteProtected_FS(uint8_t lun);
static int8_t STORAGE_Read_FS(uint8_t lun, uint8_t *buf, uint32_t blk_addr, uint16_t blk_len);
static int8_t STORAGE_Write_FS(uint8_t lun, uint8_t *buf, uint32_t blk_addr, uint16_t blk_len);
static int8_t STORAGE_GetMaxLun_FS(void);
```

F446RE_USB_MSC: usbd_storage_if.c módosítása

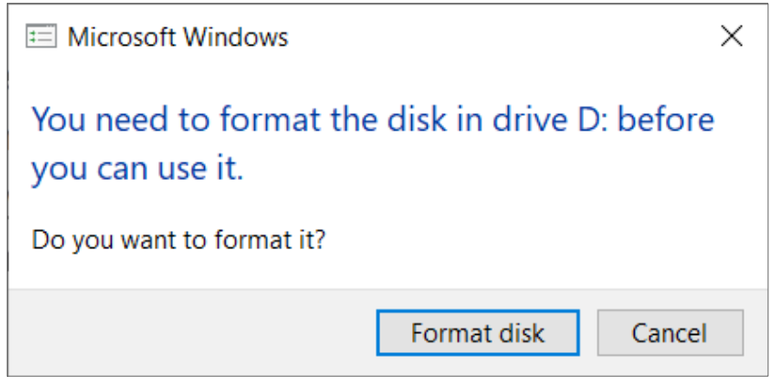
```
int8_t STORAGE_Read_FS(uint8_t lun, uint8_t *buf, uint32_t blk_addr, uint16_t blk_len)
{
  /* USER CODE BEGIN 6 */
    memcpy(buf, &buffer[blk_addr*STORAGE_BLK_SIZ],blk_len*STORAGE_BLK_SIZ); //--- copy data from RAM
    return (USBD_OK);
  /* USER CODE END 6 */
}
```

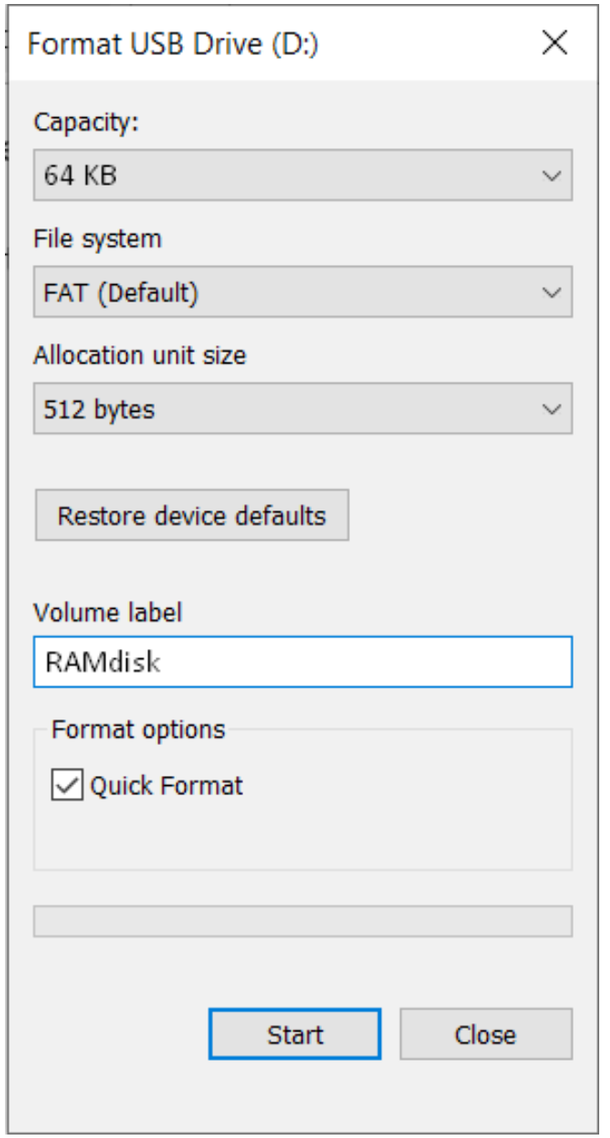
- A cím és a méret megadása blokk egységekben történik (512 bájt), ezért a tényleges cím és méret kiszámításához a blokkmérettel be kell szoroznunk.
- Az USB kapcsolaton érkező lekérések csak blokkok elővételét vagy eltárolását kérik, firmware szinten ennyiből áll a „tömegtároló” igény kiszolgálása

```
int8_t STORAGE_Write_FS(uint8_t lun, uint8_t *buf, uint32_t blk_addr, uint16_t blk_len)
{
  /* USER CODE BEGIN 7 */
    memcpy(&buffer[blk_addr*STORAGE_BLK_SIZ], buf, blk_len*STORAGE_BLK_SIZ); //--- copy data to RAM
    return (USBD_OK);
  /* USER CODE END 7 */
}
```

F446RE_USB_MSC: futási eredmény

- A főpogramon (main.c) kivételesen semmit sem változtatunk
- Fordítás/programletöltés után az első indításnál formázni kell az eszközt



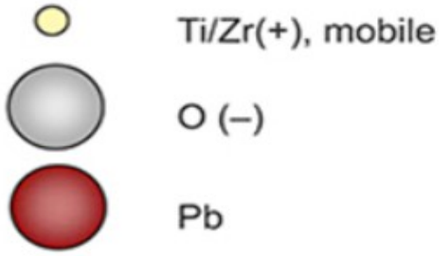


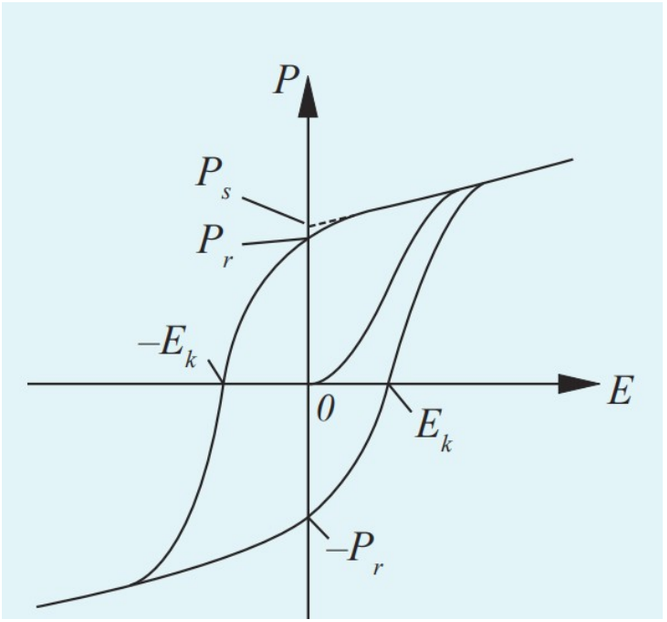
- Formázás után az új meghajtó is használható, de:
  - A kapacitása rendkívül korlátozott
  - A tartalmát minden kikapcsoláskor vagy újraindításkor elfelejti

- Miután oroszlánkörmeinket kipróbáltunk, nézzünk valami hasznosabb alkalmazást, nem felejtő tárolókkal!

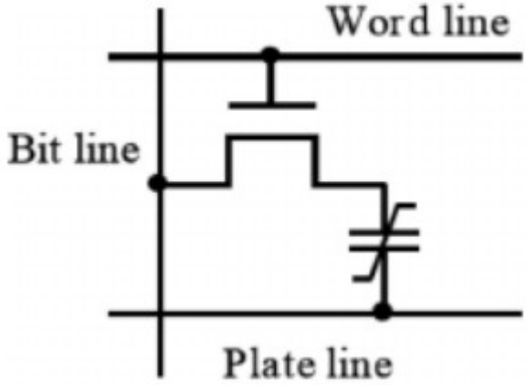
FRAM: ferroeletromos memória

- Azokat az anyagokat nevezzük **ferroelektromosnak,** amelyek spontán elektromos polarizációval rendelkeznek, melynek iránya külső elektromos mező hatására megváltoztatható.

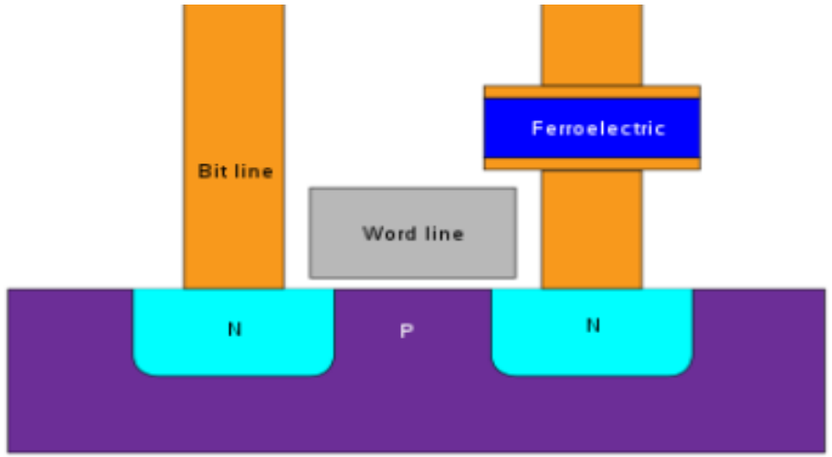




Ps – spontán polarizáció
Pr – remanens polarizáció
Ek – koercitív villamos térerősség



1T1C
Memory cell



Mikrovezérlők FRAM memóriával

- A Texas Instruments **MSP430FRxxxx** jelzésű mikrovezérlői **FRAM** programtároló memóriával vannak ellátva (a képen az **MSP430FR5994** mikrovezérlővel szerelt LaunchPad fejlesztői kártya látható)
- A **FRAM** memória előnyei
  - kis fogyasztás
  - gyors működés
  - sokszori újrairhatóság
  - adatgyűjtésre is használható

FRAM: ferroeletromos memória

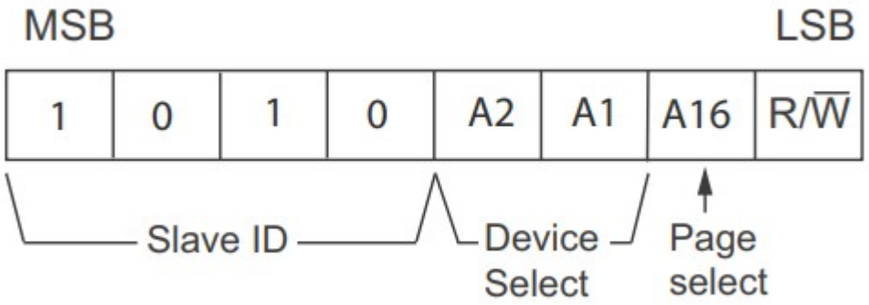
- 2008-2009 körül a **RAMTRON** nevű cég volt az éllovasa a **FRAM** memória fejlesztésének, akkori csúcstermékük 1 Mbit (128kx8) kapacitású memória volt, amelyet párhuzamos (FM28V100), **SPI** (FM25V10) és **I2C** (FM24V10) interfésszel lehetett kapni
- Mi most az **FM24V10** típust használjuk fel **USB MSC** tárként

Specifications	FRAM	SRAM	EEPROM	Flash
Non-volatile	Yes	No	Yes	Yes
Write speed (13 KB)	<10ms	<10ms	2 secs	1 sec
Average active Power (µA/MHz)	100	<60	50,000+	230
Write endurance	10^{15}	Unlimited	100,000	10,000
Bit-wise programmable	Yes	Yes	No	No
Unified Memory	Yes	No	No	No

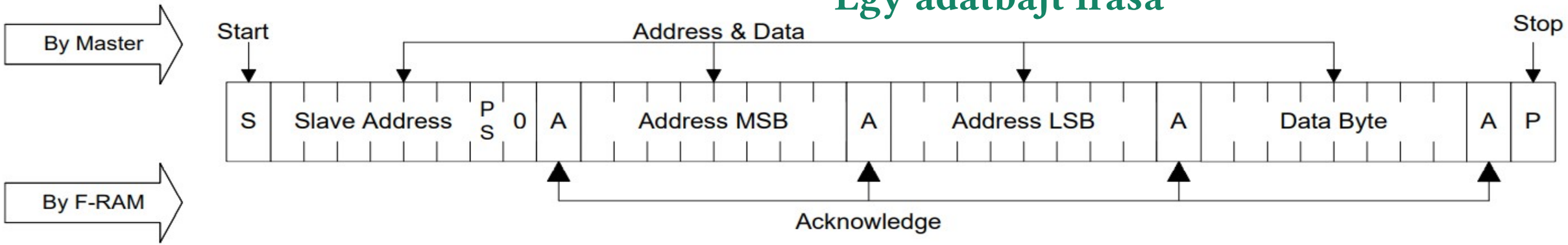
Megjegyzés:
A **RAMTRON** céget 2015-ben felvásárolta a **Cypress Semi** amelyet 2019-ben felvásárolta az **Infineon**

FRAM: FM24V10

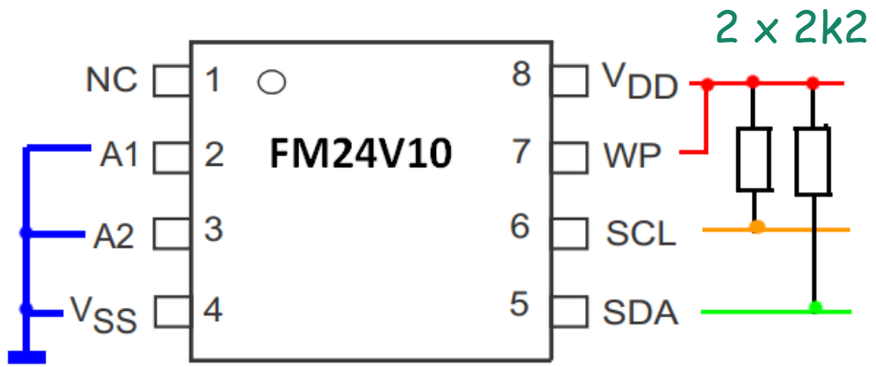
- 1-Mbit F-RAM (128K × 8)
  10^{14} újraírhatóság,
  151 év adatmegőrzés,
  késleltetés nélküli írás
- Tápfeszültség: 2.0 – 3.6 V
- I2C illesztő: 100/400 kbit/s, vagy 3.4 Mbit/s (aktív mód)
- Alacsony áramfelvétel: 175 µA @ 100 kHz,
  90 µA standby, 5 µA alvó állapotban
- Ipari kategória: –40 ℃ – 85 ℃



Egy adatbájt írása



F446RE_USB_MSC_FRAM: Kapcsolási vázlat



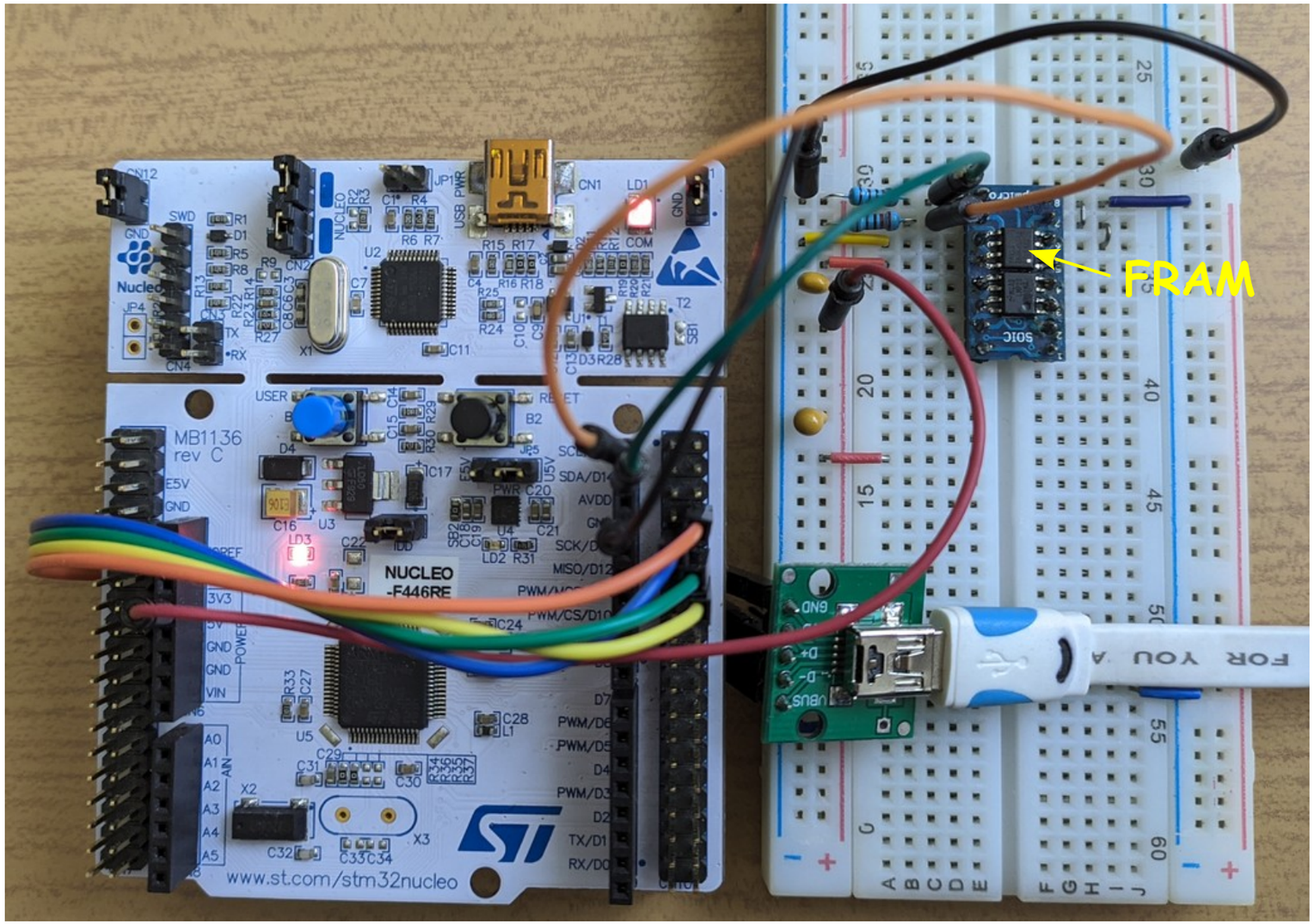
FRAM NUCLEO

SDA – PB9 (SDA)

SCL – PB8 (SCL)

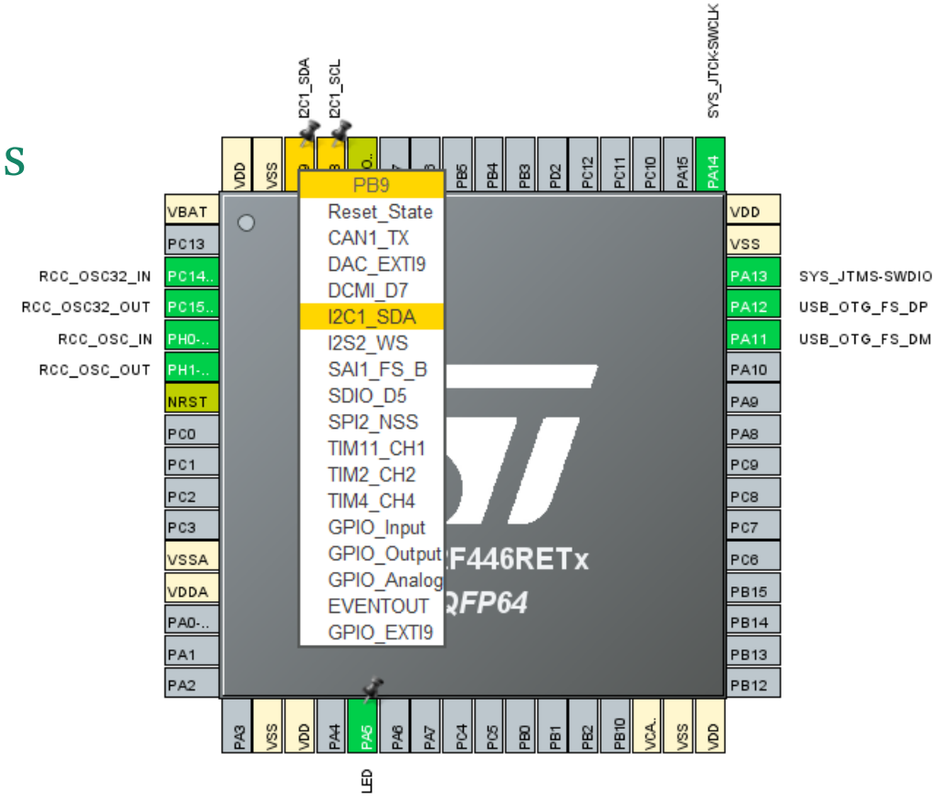
VDD – 3,3V

GND – GND



F446RE_USB_MSC_FRAM: I2C konfigurálás

- A projekt létrehozása és konfigurálása ugyanúgy történik, mint az előző programnál, csak ki kell egészítenünk az ***I2C1*** csatorna konfigurálásával
- Mivel nem az alapértelmezett **I2C** kivezetéseket használjuk ezért először a **PB9** kivezetést állítjuk **I2C1_SDA** módba
- Ezután a **PB8** kivezetést állítjuk **I2C1_SCL** módba
- Ezt követően konfigurálhatjuk az **I2C1** kommunikációs csatornát



F446RE_USB_MSC_FRAM: I2C konfigurálás

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

Categories | A->Z

- System Core
- Analog
- Timers
- Connectivity
  - CAN1
  - CAN2
  - FMPI2C1
  - I2C1
  - I2C2
  - I2C3
  - QUADSPI
  - SDIO
  - SPI1
  - SPI2
  - SPI3
  - UART4
  - UART5
  - USART1
  - USART2
  - USART3
  - USART6
  - USB_OTG_FS
  - USB_OTG_HS

I2C1 Mode and Configuration

Mode

I2C: I2C

Configuration

Reset Configuration

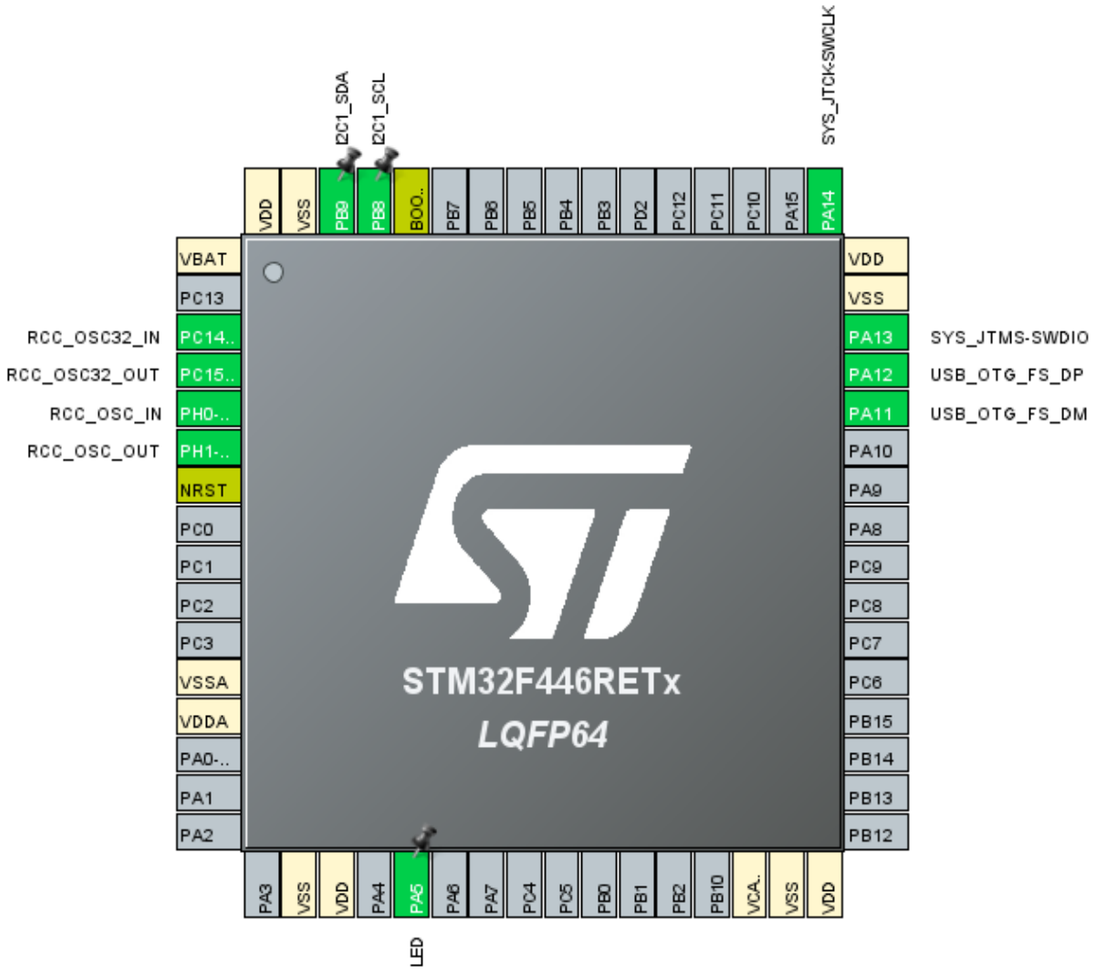
NVIC Settings | DMA Settings | GPIO Settings | Parameter Settings | User Constants

Configure the below parameters :

Search (Crtl+F)

Master Features	
I2C Speed Mode	Fast Mode
I2C Clock Speed (Hz)	400000
Fast Mode Duty Cycle	Duty cycle Tlow/Thigh = 2
Slave Features	
Clock No Stretch Mode	Disabled
Primary Address Length selection	7-bit
Dual Address Acknowledged	Disabled
Primary slave address	0
General Call address detection	Disabled

Pinout view | System view



F446RE_USB_MSC_FRAM: usbd_storage_if.c módosítása

- Kódgenerálás után csak az **usbd_storage_if.c** forrásállományt kell módosítani.
- A fájl elején módosítjuk a blokkok számát és megadjuk az I2C elérését

```
#define STORAGE_LUN_NBR                  1
#define STORAGE_BLK_NBR                  0x100     //--- 256 blokk
#define STORAGE_BLK_SIZ                  0x200     //--- blokkonként 512 bájt
/* USER CODE BEGIN PRIVATE_DEFINES */
extern I2C_HandleTypeDef hi2c1;                    //--- I2C handle
#define I2C_ADDR                         0x50      //--- I2C address (right aligned)
/* USER CODE END PRIVATE_DEFINES */
```

- Az interfész függvények közül most is csak a **STORAGE_Read_FS()** és a **STORAGE_Write_FS()** függvényeket kell átszabni

```
static int8_t STORAGE_Init_FS(uint8_t lun);
static int8_t STORAGE_GetCapacity_FS(uint8_t lun, uint32_t *block_num, uint16_t *block_size);
static int8_t STORAGE_IsReady_FS(uint8_t lun);
static int8_t STORAGE_IsWriteProtected_FS(uint8_t lun);
static int8_t STORAGE_Read_FS(uint8_t lun, uint8_t *buf, uint32_t blk_addr, uint16_t blk_len);
static int8_t STORAGE_Write_FS(uint8_t lun, uint8_t *buf, uint32_t blk_addr, uint16_t blk_len);
static int8_t STORAGE_GetMaxLun_FS(void);
```

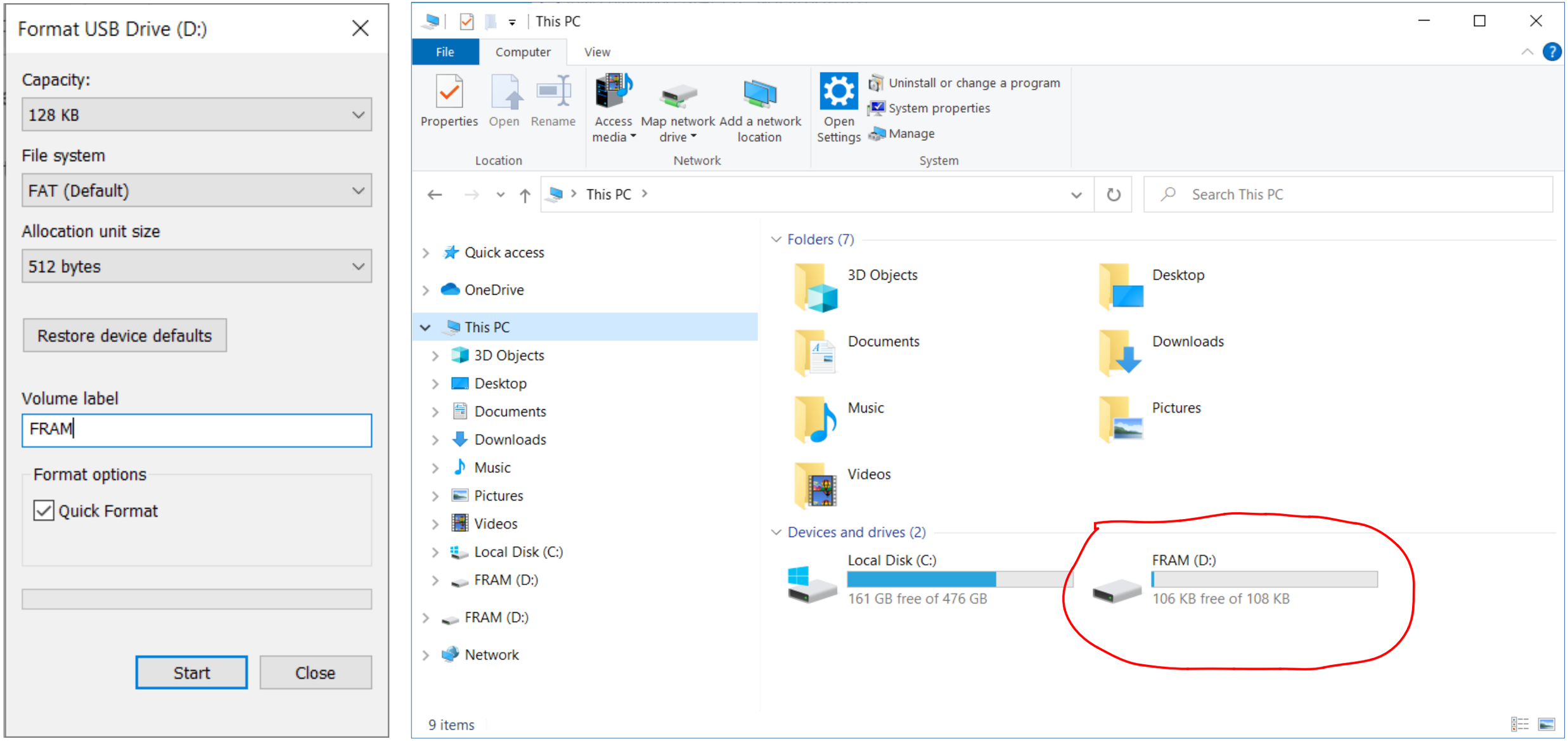
F446RE_USB_MSC_FRAM: usbd_storage_if.c módosítása

```
int8_t STORAGE_Read_FS(uint8_t lun, uint8_t *buf, uint32_t blk_addr, uint16_t blk_len) {
  uint32_t t_addr = blk_addr*STORAGE_BLK_SIZ;
  uint16_t t_len = blk_len * STORAGE_BLK_SIZ;
  uint16_t memaddr = t_addr & 0xFFFF;                  // memaddr is the lower 16 bit part (A15..A0)
  uint16_t devaddr = (0x50<<1) | ((t_addr>>15) & 2); // Combine I2C address with A16 address bit
  if(HAL_I2C_Mem_Read(&hi2c1, devaddr, memaddr, 2, buf, t_len, 100 ) == HAL_OK) return (USBD_OK);
  else return (USBD_FAIL);
}
```

- A cím és a méret megadása blokk egységekben történik (512 bájt), ezért a tényleges cím és méret kiszámításához a blokkmérettel be kell szoroznunk.
- Az USB kapcsolaton érkező lekérések csak blokkok elővételét vagy eltárolását kérik, firmware szinten ennyiből áll a „tömegtároló” igény kiszolgálása

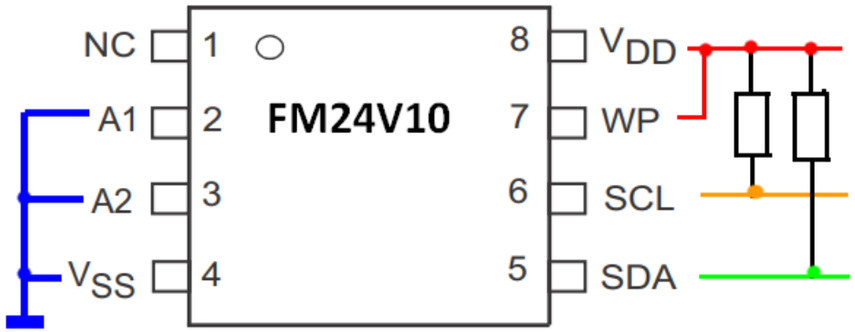
```
int8_t STORAGE_Write_FS(uint8_t lun, uint8_t *buf, uint32_t blk_addr, uint16_t blk_len) {
  uint32_t t_addr = blk_addr*STORAGE_BLK_SIZ;
  uint16_t t_len = blk_len * STORAGE_BLK_SIZ;
  uint16_t memaddr = t_addr & 0xFFFF;                  // memaddr is the lower 16 bit part (A15..A0)
  uint16_t devaddr = (0x50<<1) | ((t_addr>>15) & 2); // Combine I2C address with A16 address bit
  if(HAL_I2C_Mem_Write(&hi2c1, devaddr, memaddr, 2, buf, t_len, 100 ) == HAL_OK) return (USBD_OK);
  else return (USBD_FAIL);
}
```

F446RE_USB_MSC_FRAM: futási eredmény

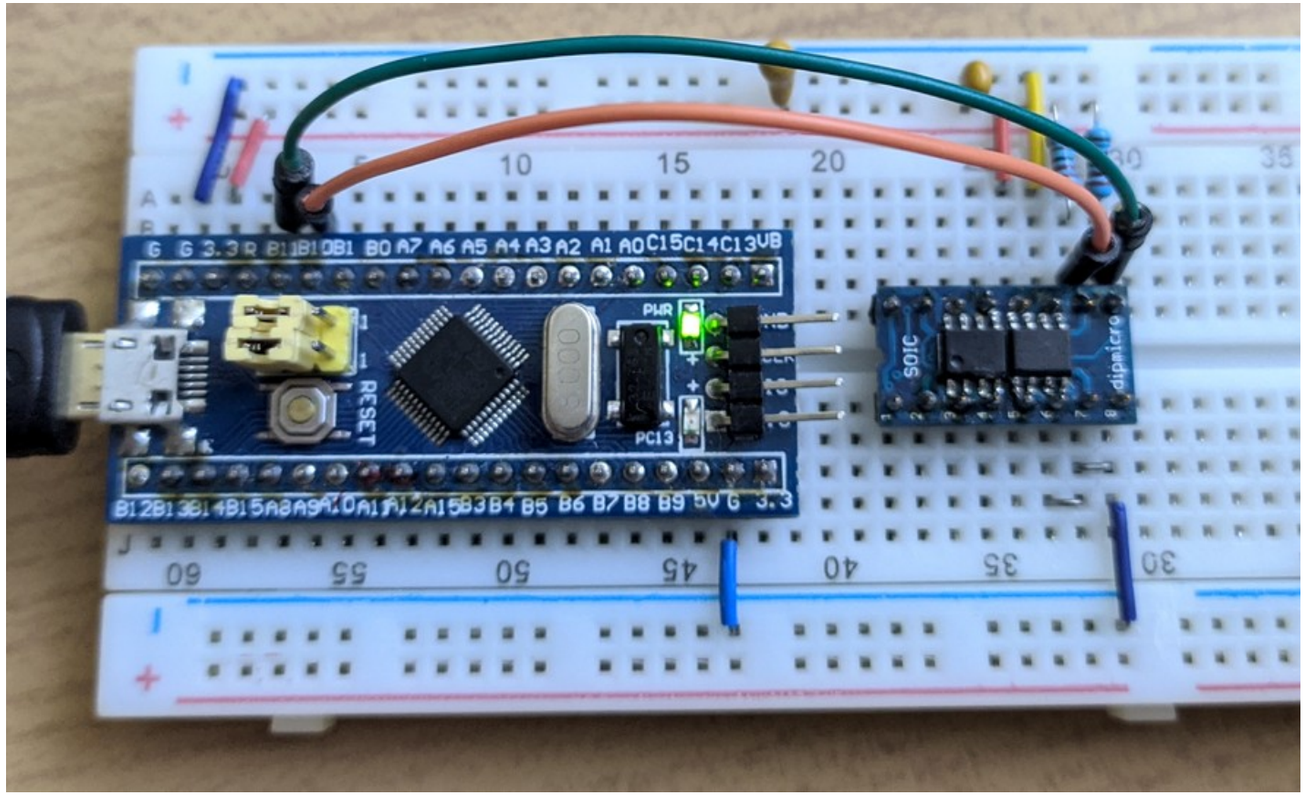


F103_USB_MSC_FRAM: bekötési vázlat

- Az **FRAM** memórián alapuló USB tömegtároló eszközt a „Blue Pill” kártyával (**STM32F103C8** mikrovezérlő) is megépíthetjük
- A különbség csupán annyi, hogy most az **I2C2** csatornát választottuk (önkényesen)
- Az **I2C** cím itt is 0x50 (+ **A16**)



SDA – PB11
SCL – PB10
VDD – 3,3V
GND – GND



F103_USB_MSC_FRAM: I2C2 konfigurálása

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

Categories | A->Z

- System Core
- Analog
- Timers
- Connectivity
  - CAN
  - I2C1
  - I2C2
  - SPI1
  - SPI2
  - USART1
  - USART2
  - USART3
  - USB
- Computing
- Middleware

I2C2 Mode and Configuration

Mode

I2C: I2C

Configuration

Reset Configuration

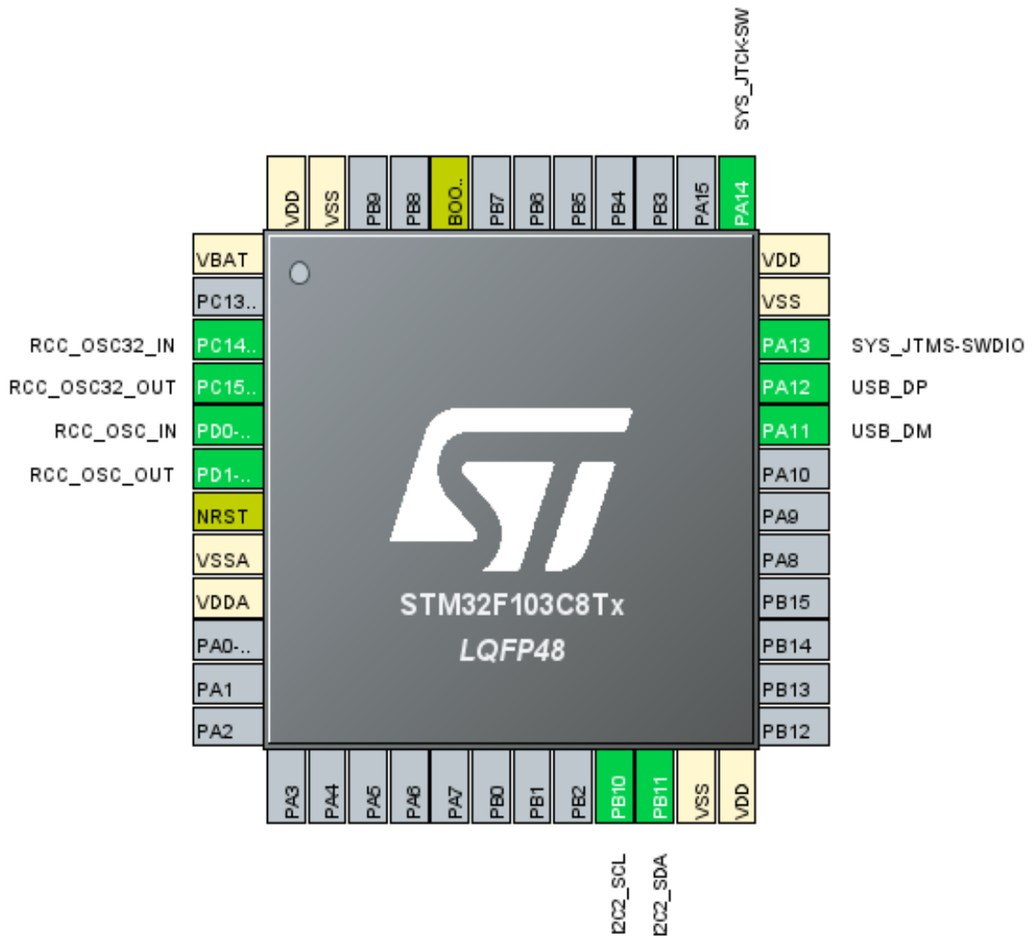
NVIC Settings | DMA Settings | GPIO Settings | Parameter Settings | User Constants

Configure the below parameters :

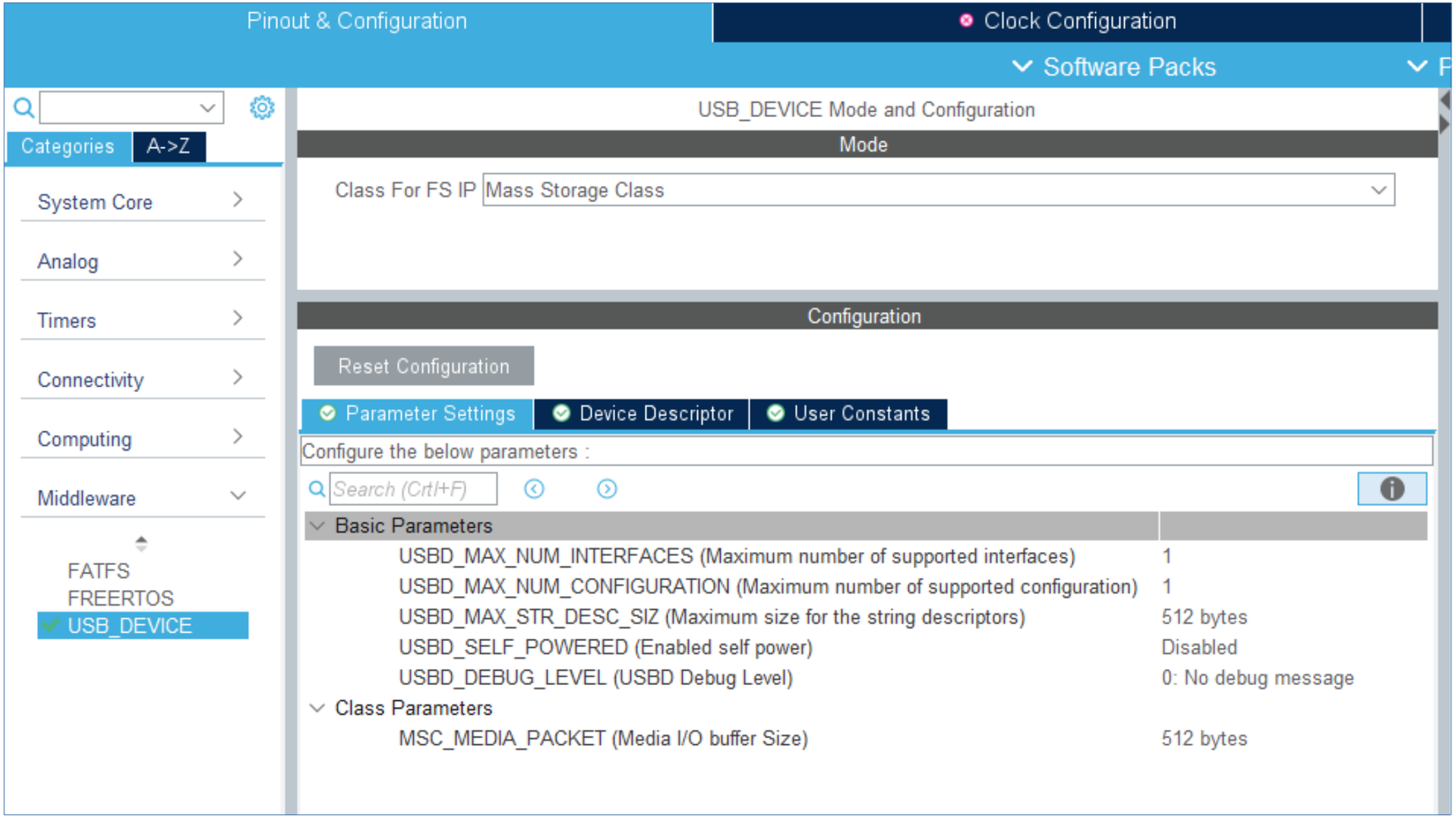
Search (Crtl+F)

Master Features	
I2C Speed Mode	Fast Mode
I2C Clock Speed (Hz)	400000
Fast Mode Duty Cycle	Duty cycle Tlow/Thigh = 2
Slave Features	
Clock No Stretch Mode	Disabled
Primary Address Length selection	7-bit
Dual Address Acknowledged	Disabled
Primary slave address	0
General Call address detection	Disabled

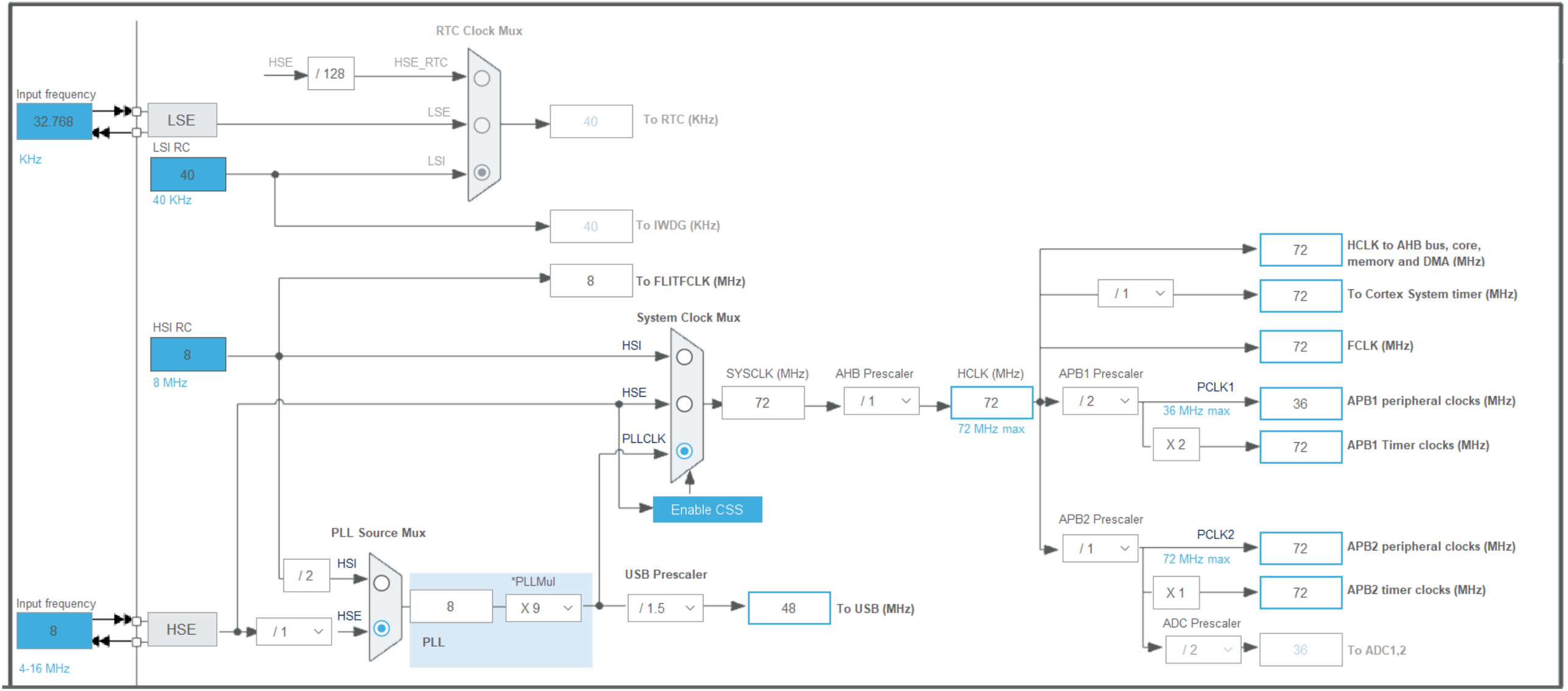
Pinout view



F103_USB_MSC_FRAM: USB konfigurálás



F103_USB_MSC_FRAM: órajelek konfigurálása



F103_USB_MSC_FRAM: usbd_storage_if.c módosítása

- Kódgenerálás után csak az **usbd_storage_if.c** forrásállományt kell módosítani.
- A fájl elején módosítjuk a blokkok számát és megadjuk az I2C elérését

```
#define STORAGE_LUN_NBR                  1
#define STORAGE_BLK_NBR                  0x100     //--- 256 blokk
#define STORAGE_BLK_SIZ                  0x200     //--- blokkonként 512 bájt
/* USER CODE BEGIN PRIVATE_DEFINES */
extern I2C_HandleTypeDef hi2c2;                    //--- I2C2 handle
#define I2C_ADDR                         0x50      //--- I2C address (right aligned)
/* USER CODE END PRIVATE_DEFINES */
```

- Az interfész függvények közül most is csak a **STORAGE_Read_FS()** és a **STORAGE_Write_FS()** függvényeket kell átszabni

```
static int8_t STORAGE_Init_FS(uint8_t lun);
static int8_t STORAGE_GetCapacity_FS(uint8_t lun, uint32_t *block_num, uint16_t *block_size);
static int8_t STORAGE_IsReady_FS(uint8_t lun);
static int8_t STORAGE_IsWriteProtected_FS(uint8_t lun);
static int8_t STORAGE_Read_FS(uint8_t lun, uint8_t *buf, uint32_t blk_addr, uint16_t blk_len);
static int8_t STORAGE_Write_FS(uint8_t lun, uint8_t *buf, uint32_t blk_addr, uint16_t blk_len);
static int8_t STORAGE_GetMaxLun_FS(void);
```

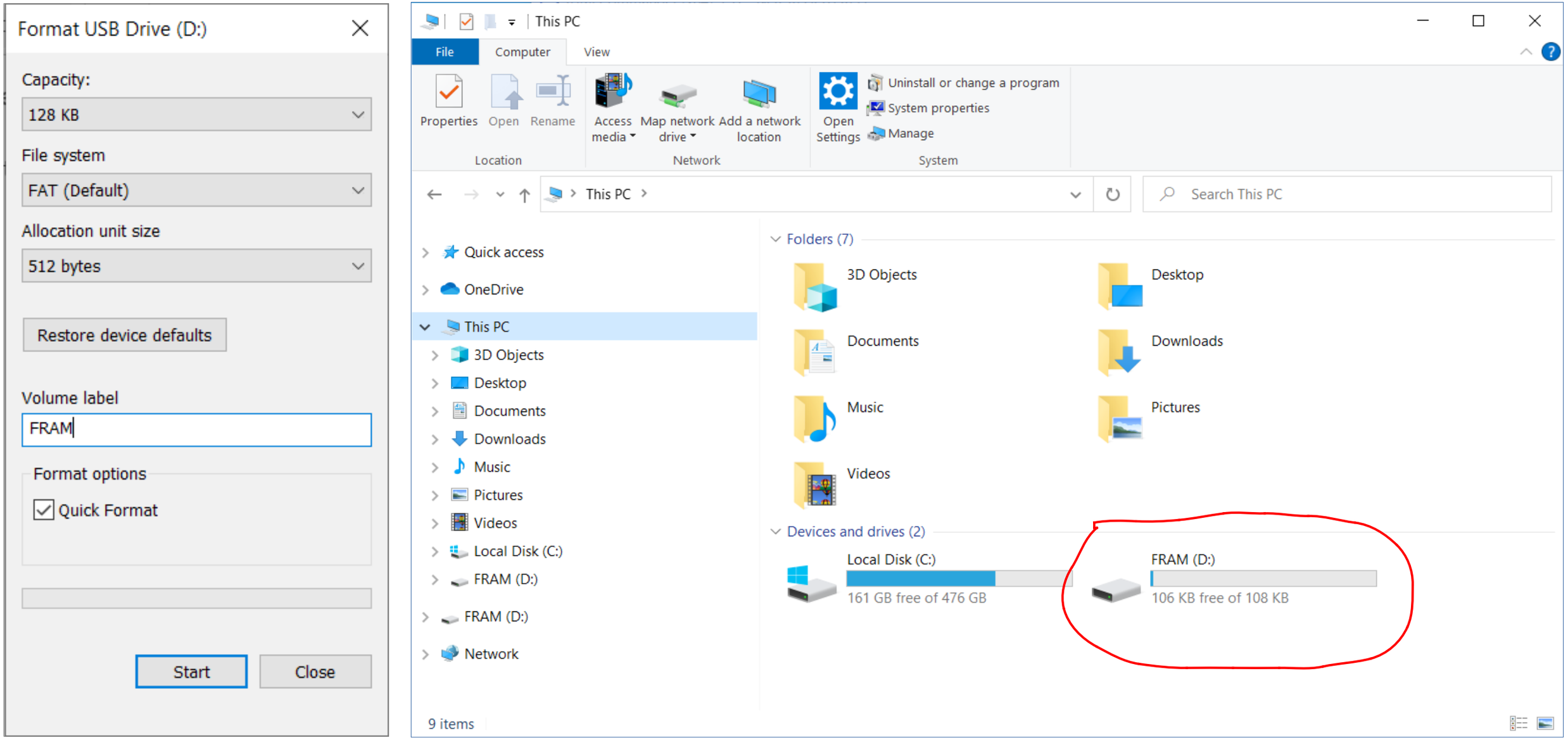
F103_USB_MSC_FRAM: usbd_storage_if.c módosítása

```
int8_t STORAGE_Read_FS(uint8_t lun, uint8_t *buf, uint32_t blk_addr, uint16_t blk_len) {
  uint32_t t_addr = blk_addr*STORAGE_BLK_SIZ;
  uint16_t t_len = blk_len * STORAGE_BLK_SIZ;
  uint16_t memaddr = t_addr & 0xFFFF;                  // memaddr is the lower 16 bit part (A15..A0)
  uint16_t devaddr = (0x50<<1) | ((t_addr>>15) & 2); // Combine I2C address with A16 address bit
  if(HAL_I2C_Mem_Read(&hi2c2, devaddr, memaddr, 2, buf, t_len, 100 ) == HAL_OK) return (USBD_OK);
  else return (USBD_FAIL);
}
```

- A cím és a méret megadása blokk egységekben történik (512 bájt), ezért a tényleges cím és méret kiszámításához a blokkmérettel be kell szoroznunk.
- Az USB kapcsolaton érkező lekérések csak blokkok elővételét vagy eltárolását kérik, firmware szinten ennyiből áll a „tömegtároló” igény kiszolgálása

```
int8_t STORAGE_Write_FS(uint8_t lun, uint8_t *buf, uint32_t blk_addr, uint16_t blk_len) {
  uint32_t t_addr = blk_addr*STORAGE_BLK_SIZ;
  uint16_t t_len = blk_len * STORAGE_BLK_SIZ;
  uint16_t memaddr = t_addr & 0xFFFF;                  // memaddr is the lower 16 bit part (A15..A0)
  uint16_t devaddr = (0x50<<1) | ((t_addr>>15) & 2); // Combine I2C address with A16 address bit
  if(HAL_I2C_Mem_Write(&hi2c2, devaddr, memaddr, 2, buf, t_len, 100 ) == HAL_OK) return (USBD_OK);
  else return (USBD_FAIL);
}
```

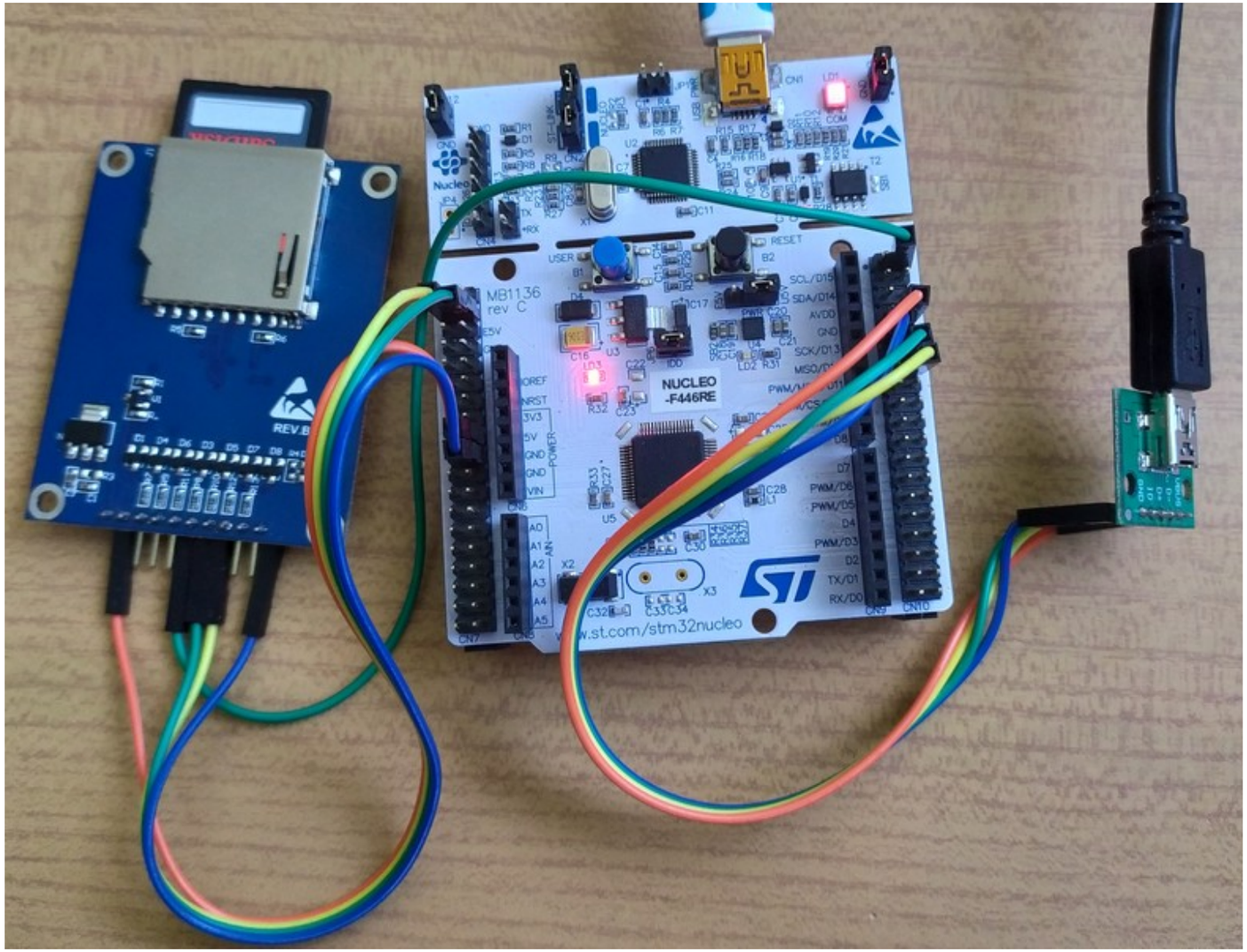
F103_USB_MSC_FRAM: futási eredmény



F446RE_USB_MSC_SDIO: kapcsolási elrendezés

- Egy 1.8” TFT kijelző modul hátulján található SD kártya foglalatot használunk fel
- A rendelkezésre álló kivezetések csak az 1-bites mód használatát teszik lehetővé (SPI vagy SDIO)
- A beépített jelszintillesztés miatt a jelszint akár 3,3V, akár 5V is lehet
- A tápfeszültség átkötéssel választható

SD breakout	Nucleo-F446RE	
SCLK	SDIO_CK	PC12
MOSI	SDIO_CMD	PD2
MISO	SDIO_D0	PC8
VCC	VCC	5V
GND	GND	GND

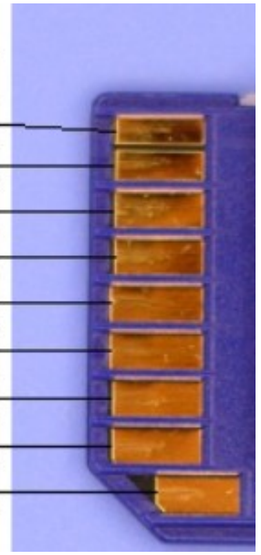


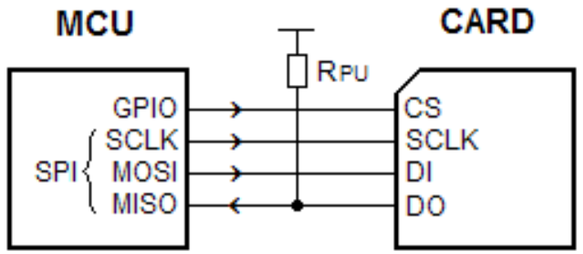
Secure Digital (SD) kártyák

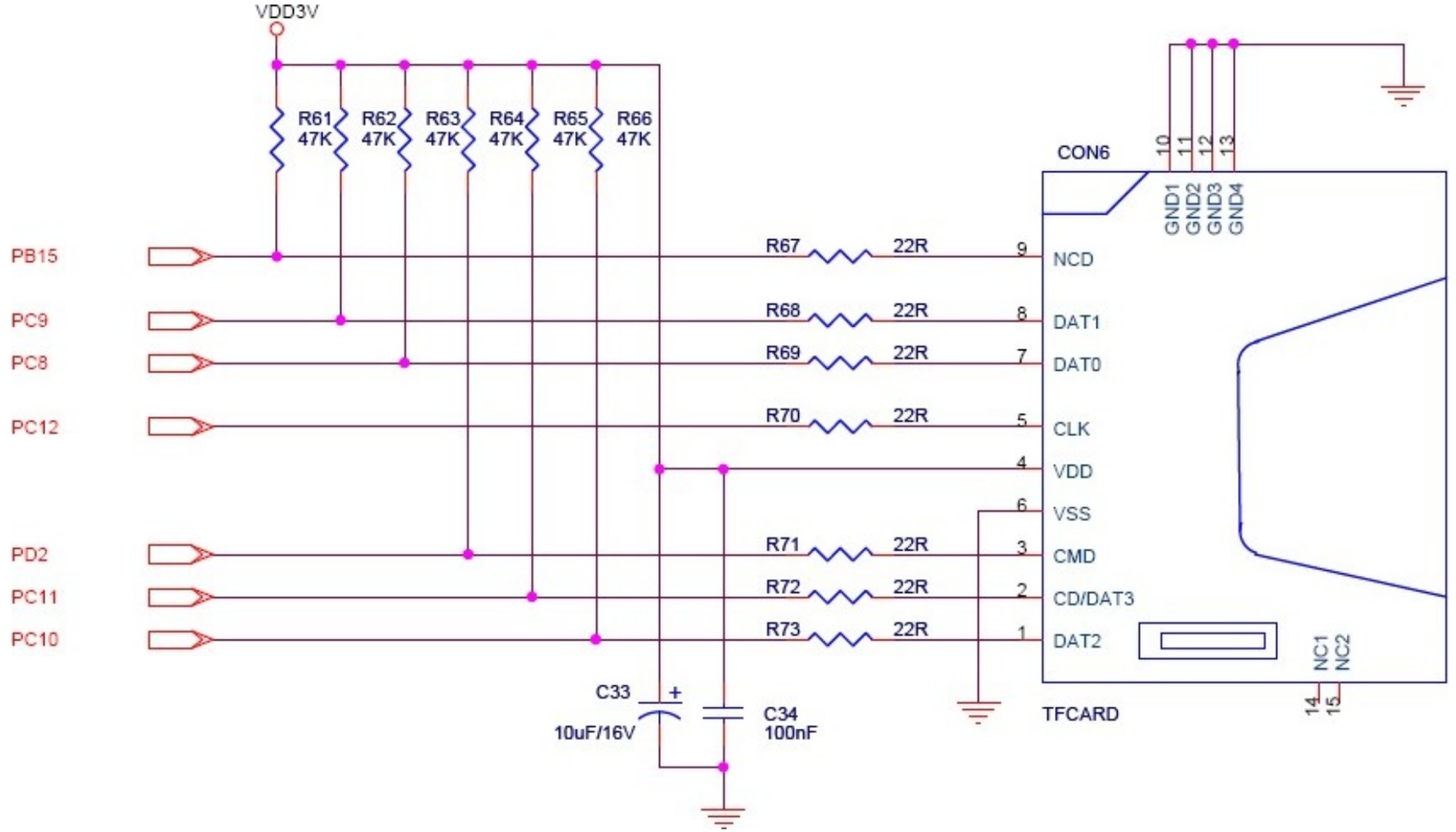
- A **Secure Digital Memory Card** *de facto* szabvány a hordozható eszközök memóriakártyái közül
- Meghajtható **SPI** módban, illetve 1, vagy 4 adatvonalas **SDIO** módban (az **STM32F446RE** mikrovezérlő rendelkezik **SDIO** perifériával is), az adatok 512 bájtos blokkokba vannak szervezve

SDC

No	SD	SPI
8	DAT1	
7	DAT0	DO
6	Vss2	
5	CLK	SCLK
4	Vcc	
3	Vss1	
2	CMD	DI
1	CAT3	CS
9	DAT2	







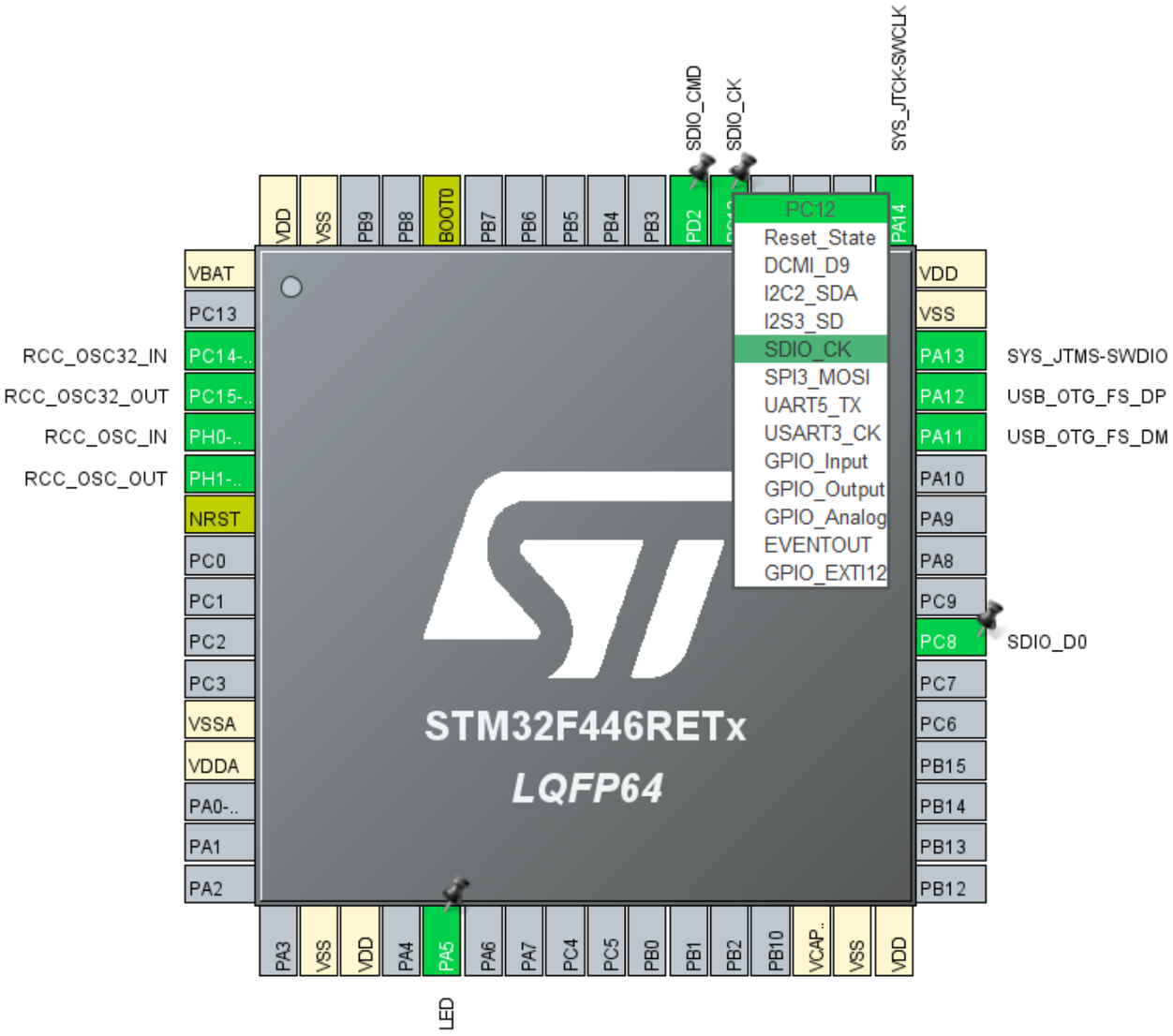
Secure Digital (SD) kártyák jelölései

- A kártya jelölései közlik a kártya kapacitását, típusát/kategóriáját, a maximális olvasási és a minimális írási sebességet
- **SD:** max. 2 GB, **SD HC:** max 32 GB (high capacity), SD XC: max 2 TB (extended capacity)

- Max. olvasási sebesség
- A kártya típusa (SD HC)
- UHS kategória (itt: UHS-I)
- SD sebességosztály (6 MB/s)
- UHS sebességosztály (itt nincs)
- Kártya kapacitása (itt 4GB)

F446RE_USB_MSC_SDIO: GPIO konfigurálás

- A projekt konfigurálását ugyanúgy csináljuk, mint az első (F446RE_USB_MSC) mintapéldánál
- Többletként csak az **SDIO** periféria konfigurálása szükséges
- Mivel nem az alapértelmezett **SDIO_CK** kivezetést használjuk, először a **PC8, PC12** és **PD2** kivezetéseket konfiguráljuk
- A kivezetések után engedélyezzük és konfiguráljuk az **SDIO** perifériát **SD 1-bit** módban!



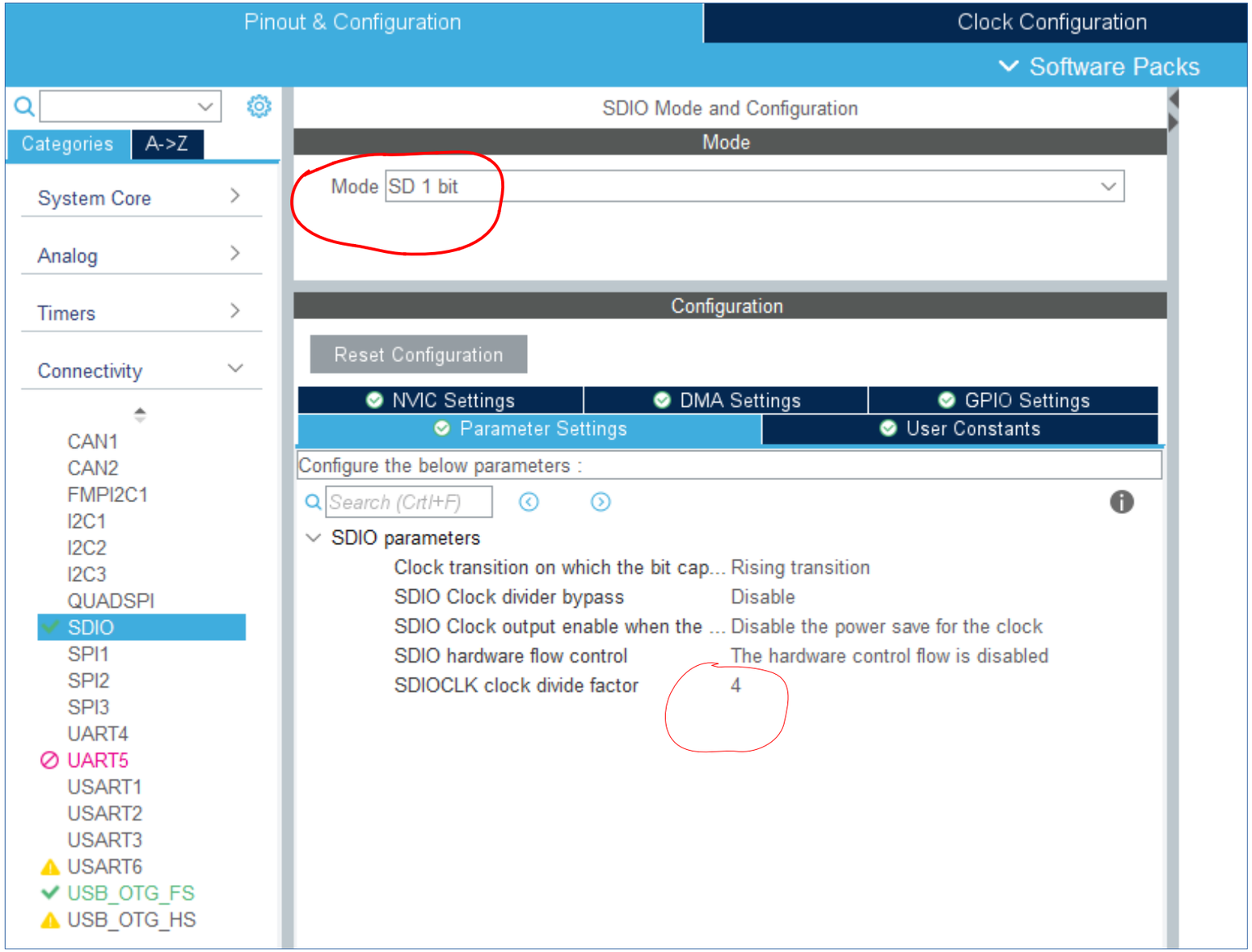
F446RE_USB_MSC_SDIO: SDIO konfigurálás

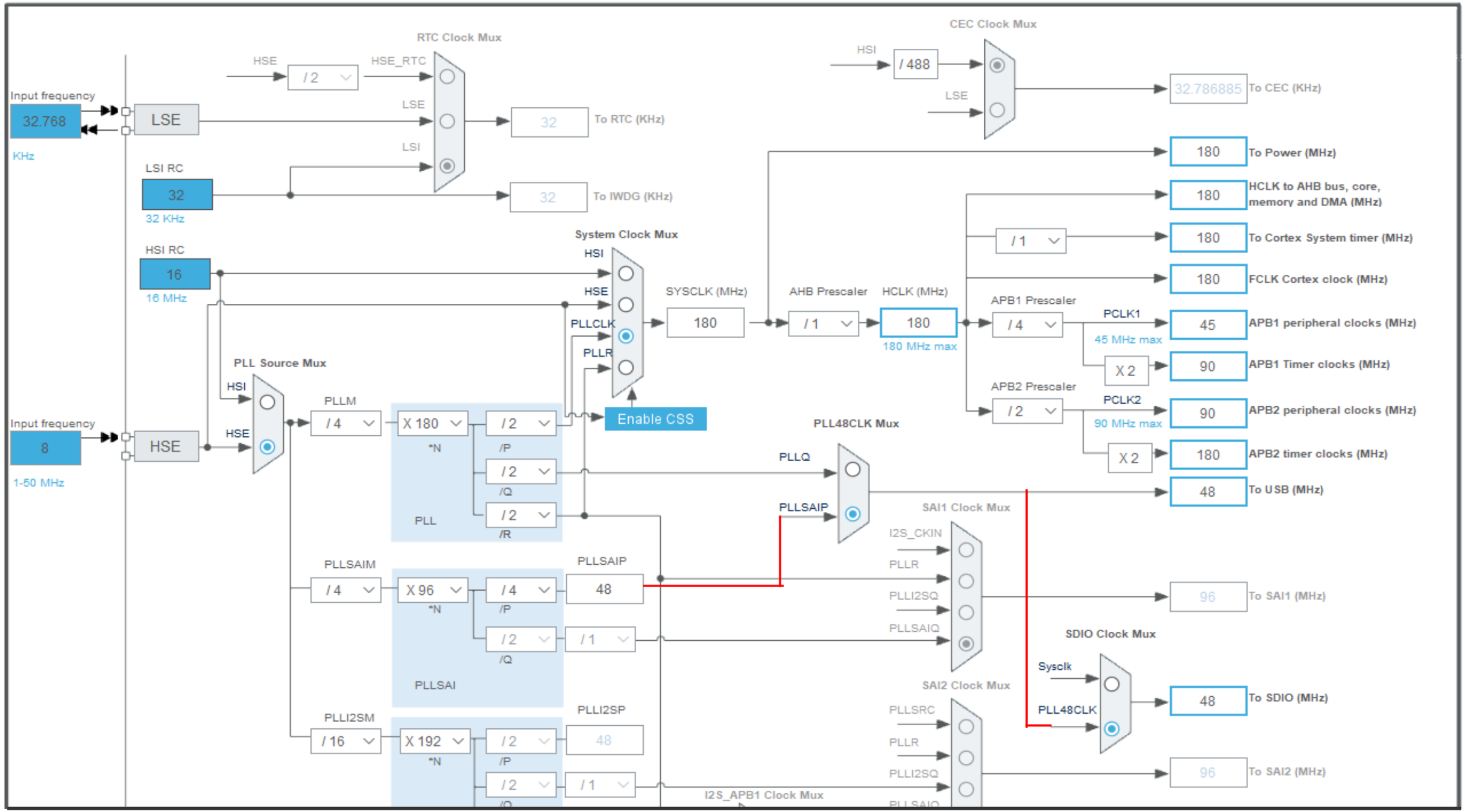
- Az **SDIO** periféria **SD 1-bit** üzemmódjának beállításán kívül fontos paraméter az órajel frekvenciája lásd a következő lapon)
- Emellett be kell állítanunk az **SDIO** bemenő órajelének leosztási arányát is (az ábrán látható leosztási arány próbálgatás eredménye – ezzel működött a program…)

$$\text{SDIO\_CK } frequency = \frac{SDIOCLK}{CLKDIV + 2}$$

esetünkben:

SDIOCLK = 48 MHz




RTC Clock Mux
HSE
/ 2
HSE_RTC
LSE
LSI
Input frequency
32.768
KHz
LSE
32
To RTC (KHz)
LSI RC
32
32 KHz
32
To IWDG (KHz)
CEC Clock Mux
HSI
/ 488
LSE
32.786885
To CEC (KHz)
System Clock Mux
HSI RC
16
16 MHz
HSI
HSE
PLLCLK
PLLR
SYSCLK (MHz)
180
AHB Prescaler
/ 1
HCLK (MHz)
180
180 MHz max
180
To Power (MHz)
180
HCLK to AHB bus, core, memory and DMA (MHz)
/ 1
180
To Cortex System timer (MHz)
180
FCLK Cortex clock (MHz)
APB1 Prescaler
/ 4
PCLK1
45 MHz max
45
APB1 peripheral clocks (MHz)
X 2
90
APB1 Timer clocks (MHz)
APB2 Prescaler
/ 2
PCLK2
90 MHz max
90
APB2 peripheral clocks (MHz)
X 2
180
APB2 timer clocks (MHz)
PLL Source Mux
HSI
HSE
Input frequency
8
1-50 MHz
HSE
PLLM
/ 4
X 180
*N
/ 2
/P
/ 2
/Q
/ 2
/R
PLL
Enable CSS
PLL48CLK Mux
PLLQ
PLLSAIP
48
To USB (MHz)
SAI1 Clock Mux
I2S_CKIN
PLLR
PLLI2SQ
PLLSAIQ
96
To SAI1 (MHz)
PLLSAIM
/ 4
X 96
*N
/ 4
/P
PLLSAIP
48
/ 2
/Q
/ 1
PLLSAI
SDIO Clock Mux
Sysclk
PLL48CLK
48
To SDIO (MHz)
SAI2 Clock Mux
PLLSRC
PLLR
PLLI2SQ
PLLSAIQ
96
To SAI2 (MHz)
PLLI2SM
/ 16
X 192
*N
/ 2
/P
PLLI2SP
48
/ 2
/Q
/ 1
I2S_APB1 Clock Mux

F446RE_USB_MSC_SDIO: usbd_storage_if.c módosítása

- Kódgenerálás után csak az **usbd_storage_if.c** forrásállományt kell módosítani, ezúttal a ControllersTech: STM32 USB MSC cikk útmutatását követjük

```
#define STORAGE_LUN_NBR                  1
#define STORAGE_BLK_NBR                  0x10000   //--- Don't care (not used)
#define STORAGE_BLK_SIZ                  0x200

/* USER CODE BEGIN PRIVATE_DEFINES */
extern SD_HandleTypeDef hsd;                       //--- SDIO handle
/* USER CODE END PRIVATE_DEFINES */
```

- Az interfész függvények közül **STORAGE_Read_FS()** és **STORAGE_Write_FS()** mellett a **STORAGE_GetCapacity_FS()** függvényt is át kell szabni

```
static int8_t STORAGE_Init_FS(uint8_t lun);
static int8_t STORAGE_GetCapacity_FS(uint8_t lun, uint32_t *block_num, uint16_t *block_size);
static int8_t STORAGE_IsReady_FS(uint8_t lun);
static int8_t STORAGE_IsWriteProtected_FS(uint8_t lun);
static int8_t STORAGE_Read_FS(uint8_t lun, uint8_t *buf, uint32_t blk_addr, uint16_t blk_len);
static int8_t STORAGE_Write_FS(uint8_t lun, uint8_t *buf, uint32_t blk_addr, uint16_t blk_len);
static int8_t STORAGE_GetMaxLun_FS(void);
```

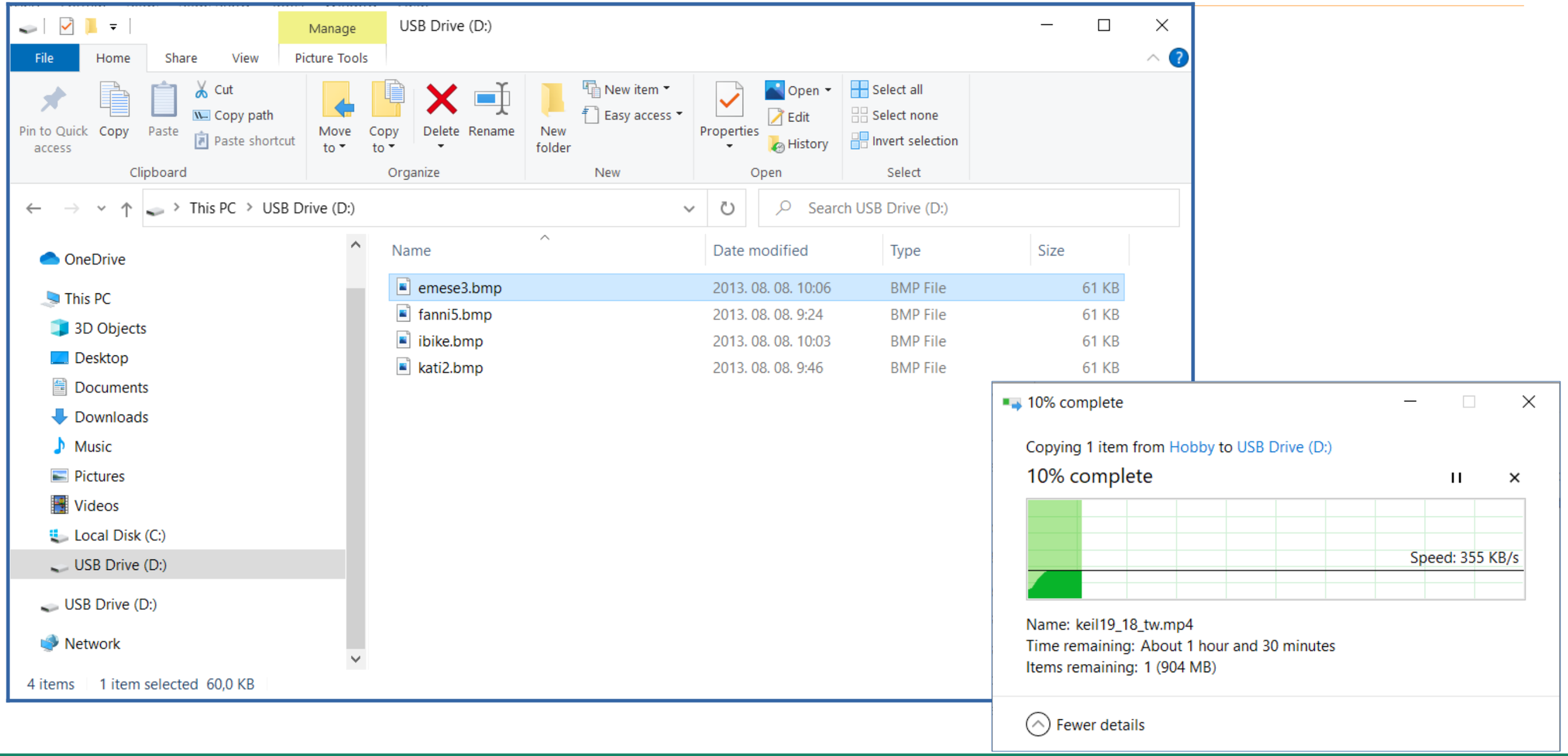
F446RE_USB_MSC_SDIO: usbd_storage_if.c módosítása

```
int8_t STORAGE_GetCapacity_FS(uint8_t lun, uint32_t *block_num, uint16_t *block_size) {
  HAL_SD_CardInfoTypeDef info;
  HAL_SD_GetCardInfo(&hsd, &info);        //--- Read number and size of data blocks from the card
  *block_num =  info.LogBlockNbr  - 1;
  *block_size = info.LogBlockSize;
  ret = USBD_OK;
}
```

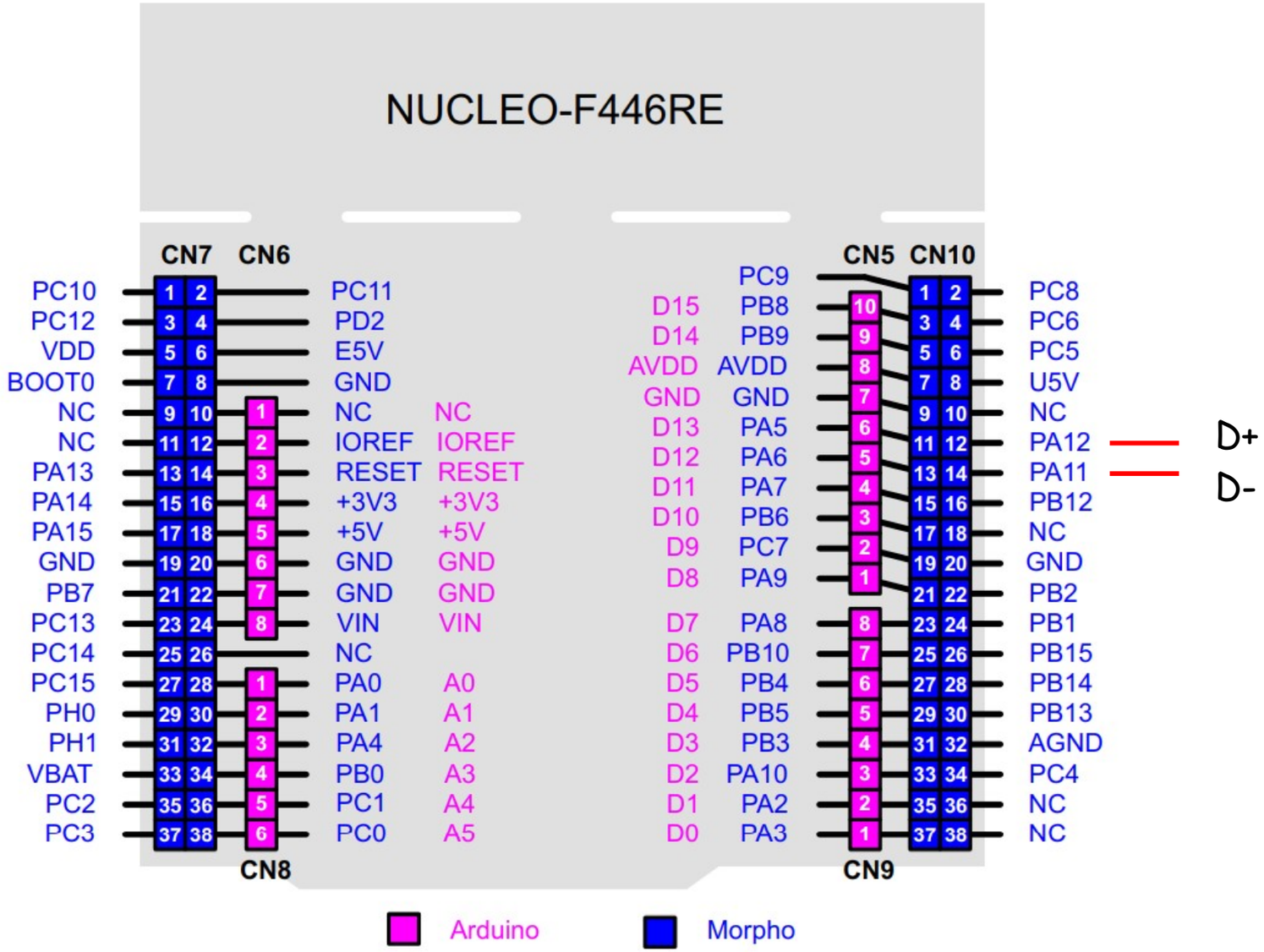
```
int8_t STORAGE_Read_FS(uint8_t lun, uint8_t *buf, uint32_t blk_addr, uint16_t blk_len) {
    HAL_SD_ReadBlocks(&hsd, buf, blk_addr, blk_len, HAL_MAX_DELAY);
    /* Wait until SD card is ready to use for new operation */
    while (HAL_SD_GetCardState(&hsd) != HAL_SD_CARD_TRANSFER){}
    return (USBD_OK);
}
```

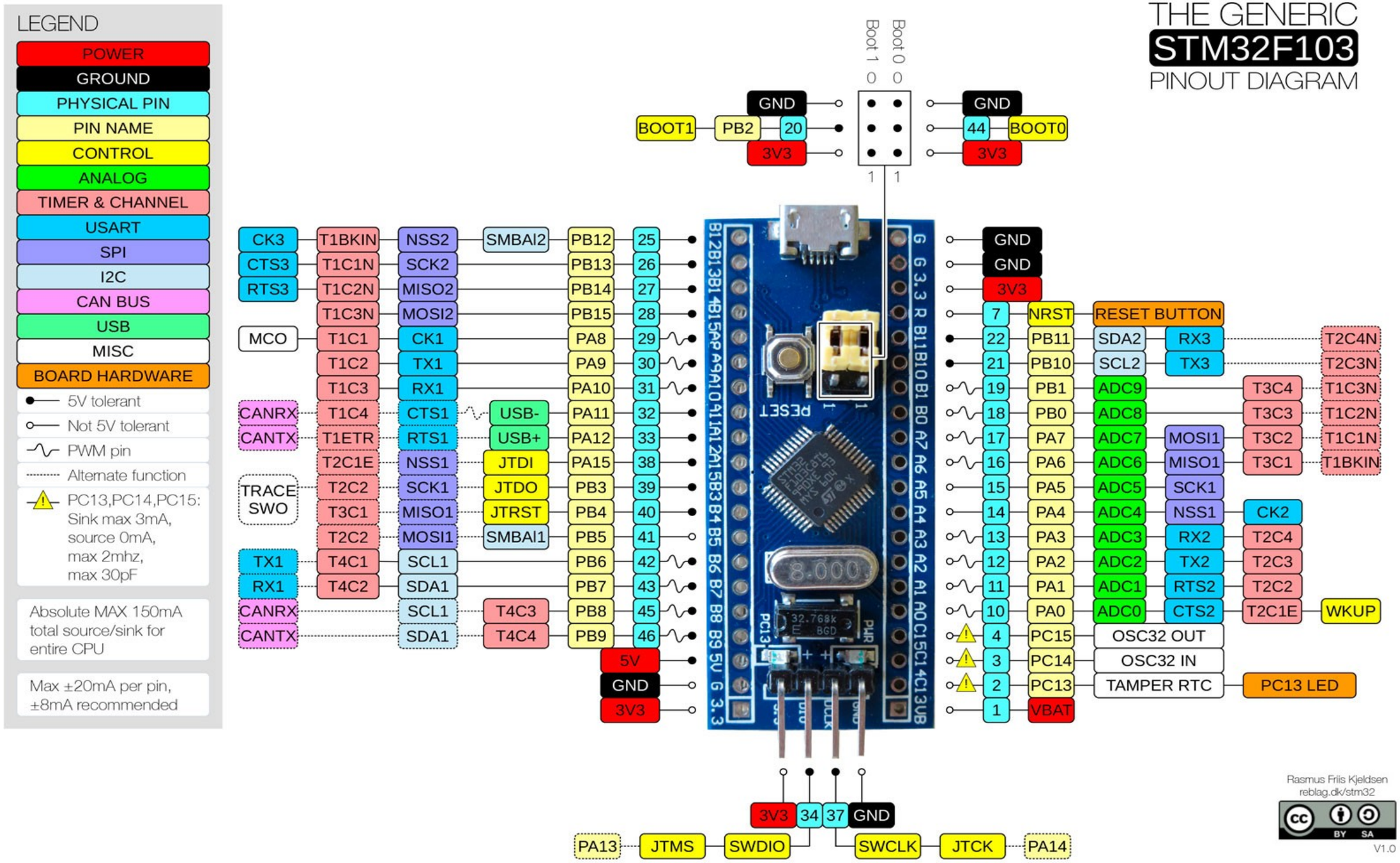
```
int8_t STORAGE_Write_FS(uint8_t lun, uint8_t *buf, uint32_t blk_addr, uint16_t blk_len) {
    HAL_SD_WriteBlocks(&hsd, buf, blk_addr, blk_len, HAL_MAX_DELAY);
    /* Wait until SD card is ready to use for new operation */
    while (HAL_SD_GetCardState(&hsd) != HAL_SD_CARD_TRANSFER){}
    return (USBD_OK);
}
```

F446RE_USB_MSC_SDIO: futási eredmény



NUCLEO-F446RE kivezetések




LEGEND
POWER
GROUND
PHYSICAL PIN
PIN NAME
CONTROL
ANALOG
TIMER & CHANNEL
USART
SPI
I2C
CAN BUS
USB
MISC
BOARD HARDWARE
5V tolerant
Not 5V tolerant
PWM pin
Alternate function
PC13,PC14,PC15: Sink max 3mA, source 0mA, max 2mhz, max 30pF
Absolute MAX 150mA total source/sink for entire CPU
Max ±20mA per pin, ±8mA recommended
THE GENERIC
STM32F103
PINOUT DIAGRAM
Boot 1
Boot 0
GND
BOOT1
PB2
20
3V3
GND
44
BOOT0
3V3
CK3
CTS3
RTS3
MCO
CANRX
CANTX
TRACE SWO
TX1
RX1
CANRX
CANTX
T1BKIN
T1C1N
T1C2N
T1C3N
T1C1
T1C2
T1C3
T1C4
T1ETR
T2C1E
T2C2
T3C1
T2C2
T4C1
T4C2
NSS2
SCK2
MISO2
MOSI2
CK1
TX1
RX1
CTS1
RTS1
NSS1
SCK1
MISO1
MOSI1
SCL1
SDA1
SCL1
SDA1
SMBAI2
USB-
USB+
JTDI
JTDO
JTRST
SMBAI1
T4C3
T4C4
PB12
PB13
PB14
PB15
PA8
PA9
PA10
PA11
PA12
PA15
PB3
PB4
PB5
PB6
PB7
PB8
PB9
25
26
27
28
29
30
31
32
33
38
39
40
41
42
43
45
46
5V
GND
3V3
GND
GND
3V3
7
22
21
19
18
17
16
15
14
13
12
11
10
4
3
2
1
NRST
PB11
PB10
PB1
PB0
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
PC15
PC14
PC13
VBAT
RESET BUTTON
SDA2
SCL2
ADC9
ADC8
ADC7
ADC6
ADC5
ADC4
ADC3
ADC2
ADC1
ADC0
OSC32 OUT
OSC32 IN
TAMPER RTC
RX3
TX3
MOSI1
MISO1
SCK1
NSS1
RX2
TX2
RTS2
CTS2
T3C4
T3C3
T3C2
T3C1
CK2
T2C4
T2C3
T2C2
T2C1E
PC13 LED
T2C4N
T2C3N
T1C3N
T1C2N
T1C1N
T1BKIN
WKUP
3V3
34
37
GND
PA13
JTMS
SWDIO
SWCLK
JTCK
PA14
Rasmus Friis Kjeldsen
reblag.dk/stm32
V1.0